mikroKLAN 2

P. 1877/84

2 1984

informatyka

Kody kreskowe
Przetwarzanie obrazów
COMPCONTROL'83

WYDAWNICTWO NOT SIGMA

informatyka

2 1984

Nr 2
Miesięcznik Rok XIX
Luty 1984

Organ Komitetu Informatyki MNSzWiT oraz Komitetu Naukowo-Technicznego NOT ds. Informatyki

KOLEGIUM REDAKCYJNE:

Mgr inż. Zbigniew GLUZA, mgr Teresa JABŁOŃSKA (sekretarz), Władysław KLEPACZ (zastępca redaktora naczelnego), prof. dr hab. Leon ŁUKASZEWICZ (redaktor naczelny), mgr inż. Andrzej PIOTROWSKI, mgr Marek SOBCZYK, mgr Andrzej SZAŁAS, dr inż. Janusz ZALEWSKI

STALE WSPÓŁPRACUJĄ:

Mgr Adam B. EMPACHER, dr Janusz GWIAZDA (Libia), dr Jacek OWCZARCZYK, dr Jakub TATARKIEWICZ, mgr inż. Teresa WILCZEK

PRZEWODNICZĄCY RADY PROGRAMOWEJ:

Prof. dr hab. Tadeusz PECHE

Materiałów nie zamówionych redakcja nie zwraca

Redakcja: 00-041 Warszawa, ul. Jasna 14/16, pok. 243 i 244, tel. 27-71-40 lub 26-82-61 w. 184

Zakł. Graf. „Tamka". Zam. 2377. Obj. 4,0 ark. druk. Nakład 3650 egz. T-56.

INDEKS 36124

Cena egzemplarza zł 75,—
Prenumerata roczna zł 900,—

WYDAWNICTWO SIGMA

NACZELNA ORGANIZACJA TECHNICZNA

CZASOPISMA I KSIĄŻKI TECHNICZNYCH

W NUMERZE:

Strona

Interakcyjny system przetwarzania obrazów
Jacek Owczarczyk 2

MODULA-2 — język lat osiemdziesiątych (3). Opis składni
Piotr Fuglewicz 6

Programowy tester jednostek pamięci na dyskach elastycznych PL x 45D.5
Ewa Serafin, Wiesław Serafin 10

Przegląd języków wysokiego poziomu (4). Języki nowoczesne
Oprac. Teresa Wójciekian, Halina Ciechomska 12

mikroKLAN 13

— Rozmowa z Leszkiem Wilkiem
— Programowa obsługa pamięci kasetowej
— Stałe pułapki
— FORTH (2)
— Procedura „Wielkanoc"
— Krzemowy CP/M
— Kopiowanie ekranu na drukarkę graficzną

ZE ŚWIATA 24

— COMPCONTROL'83
— Dotykać komputer
— MEDINFO'83
— EUROMICRO

LISTY 29

— „Przegląd mikroprocesorów 16-bitowych"

RECENZJE 30

— Kalendarz informatyczny

TERMINOLOGIA 31

— Cykl życia oprogramowania

CZWARTA OKŁADKA
Andrzej Piątka

W NAJBLIŻSZYCH NUMERACH:

- Marek Sobczyk i Andrzej Szałas o PROLOGU
- Józef B. Lewoc o perspektywach Jednolitego Systemu
- Jacek Mirkowski i Adam Piątkowski o laboratorium dydaktycznym z rozłożoną inteligencją
- Jakub Tatarkiewicz o sprzęgu HP-IL
- Witold Abramowicz o MODULI-2 i LILITH
- Roman Żelazny o narzędziach inżynierii oprogramowania
- Ryszard K. Kott i Danuta Magdzik o technikach interpretacji dla mikrokomputerów
- Cezary Zieliński i Teresa Wilczek o robotyce

P. 1877/84

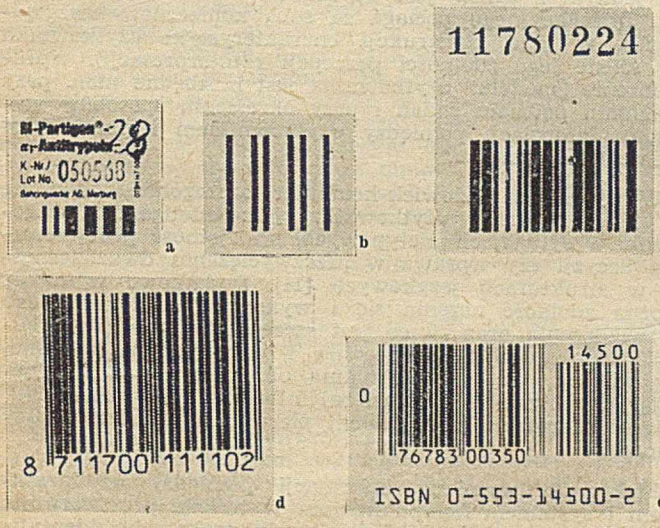
Różdżka i kreski

Tytułowa różdżka nie ma nic wspólnego z radiostezją (jak na razie radiesteci nie stosują informatyki i vice versa...). Jest to próba spolszczenia angielskiego słowa „wand", którym to słowem określa się w literaturze fachowej czytniki optyczne. Czytające..., no właśnie — co? Dosłowne tłumaczenie oznacza kody kreskowe (ang. Bar Code) (bynajmniej nie belkowe, jak mi się to zdarzyło przeczytać w książce M. Kozarskiego i Z. Szurmaka „Minikalkulatory", WNT 1980). W każdym razie pałeczka taka jest rzeczywiście magiczna, bo jednym pociągnięciem może odczytać nie tylko skomplikowany kod, ale nawet wiele linii programu komputerowego .

Chciałbym przybliżyć nieco Czytelnikom metody kodowania oraz odczytywania kodów kreskowych, a także postawić pytanie, czy powinniśmy już w Polsce przyjąć jakiś standard kodowania, a także kiedy i gdzie go wprowadzać. Jako pretekst do rozważań niech posłuży pierwsze światowe seminarium poświęcone technice automatycznego identyfikowania SCAN-TECH'83. Seminarium to odbyło się w San Diego (USA) w dniach 25—28 września 1983.

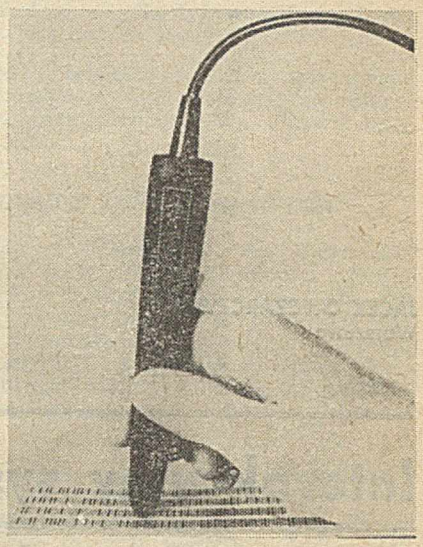
Technika kodowania kreskowego rozpowszechniła się, gdyż jest ona zdecydowanie tańsza od automatycznego odczytu cyfr i innych znaków. Dla porównania: typowy czytnik optyczny znaków kosztuje w tej chwili na świecie ok. 1000 dol., gdy tymczasem czytnik kodów kreskowych tylko 200 dol. Oczywiście, kody kreskowe mają też jedną, ale dla nas zasadniczą wadę: nie nadają się do odczytu przez człowieka. Tak więc w praktyce łączy się z reguły kod kreskowy z zapisem liczby — (p. fot. 1).

Najprostsza konstrukcja kodu polega na wyborze tzw. modułu, czyli podstawowej jednostki grubości. Przykładowo — może ona odpowiadać cienkiej kresce. Może też odpowiadać przerwie między kreskami. Aby kodowanie było jednoznaczne, potrzebujemy jeszcze jednego symbolu. Może być nim grubsza-kreska (o grubości kilku modułów). Najprostszy kod dwójkowy tworzymy interpetując szerokie kreski jako 1, a wąskie kreski — jako 0. Kreski są oddzielone zawsze wąską przerwą. Czytnik emituje wiązkę światła (nie musi to być światło laserowe, w zupełności wystarczy dioda świecąca), która ulegając odbiciu od jasnego papieru daje sygnał. Jednocześnie ciemne kreski dają o wiele słabsze odbicie. W ten sposób czytnik może odróżniać przerwy od kresek. Porównując czas odczytu kolejnych elementów ze wskazaniami wewnętrznego zegara, czytnik odróżnia też wąskie i szerokie kreski oraz uniezależnia się od prędkości przesuwania. Błędem najczęściej popełnianym przez początkujących użytkowników jest zbyt wolne przesuwanie czytnika — typowa prędkość mieści się w zakresie 10—100 cm/s.



Fot. 1. Przykłady kodów kreskowych wykorzystywanych na świecie a) RFN, b) Poczta USA, c) Japonia, d) EAN (European Article Number) — Belgia, e) UPC (Universal Product Code) — USA wraz z kodem ISBN

Istnieją dwa rodzaje czytników: ręczne oraz stacjonarne. Ręczne służą głównie do obsługi urządzeń przenośnych, natomiast stacjonarne (dla ułatwienia pracy — z reguły wyposażone w laserowy system badający wiele kierunków) stanowią głównie wyposażenie kas w sklepach. Ponadto czytniki używane są wszędzie tam, gdzie zachodzi potrzeba identyfikowania dużych ilości różnych przedmiotów — a więc w magazynach, halach produkcyjnych, aptekach, bibliotekach.



Fot. 2. Czytnik HP 82153A Wand do komputera HP41C

Na zdjęciu 1 pokazano niewielki wybór różnych typów kodów kreskowych. Niewielki, gdyż został on zebrany z różnych opakowań dostępnych w domu autora. To chyba najlepiej świadczy o rozpowszechnieniu się tych kodów, bo przecież w Polsce jeszcze się ich nie stosuje.

Przepisy amerykańskie oraz zachodnioeuropejskie wymagają już od kilku lat, by wszystkie wyroby były oznaczane kodami kreskowymi. Ponadto wiele firm stosuje takie kody w celu ułatwienia sobie pracy. Stąd wielka różnorodność typów kodowania, szerokości kresek i przerw. Na szczęście niektórzy producenci czytników kodów kreskowych (np. Hewlett-Packard model HP82174A) oferują już czytniki przeznaczone do dekodowania różnorodnych systemów. Jak dotąd tylko jeden typ komputera (HP41C) wyposażony został w czytnik kodów optycznych, który umożliwia wprowadzanie programów zapisanych w kodzie kreskowym. Ten sposób zapisu programów i ich powielania poprzez kopiowanie kserograficzne wydaje się mieć dużą przyszłość, w szczególności dla posiadaczy komputerów osobistych. Być może już niedługo INFORMATYKA będzie publikowała algorytmy jako serie kresek...

Czy w Polsce istnieją przesłanki dla stosowania kodów kreskowych? Niewątpliwie koniecznością jest najbardziej typowe zastosowanie do oznaczania produktów sprzedawanych masowo. Nie myślę tu tylko o eksporcie (gdzie odbiorcy już za parę lat mogą żądać takiego oznaczenia towarów), ale także o olbrzymim ułatwieniu pracy wszelkich magazynów i rozdzielni towarów. Drugim obszarem rozpowszechnienia kodów kreskowych stały się na świecie biblioteki. Skoro możemy w Polsce każdą książkę wyposażać w numer ISBN (międzynarodowa symbolizacja bibliograficzna), to możemy chyba też drukować na okładce ten sam kod w postaci kreskowej.

Jeszcze problem czytników. Nie sądzę, byśmy szybko wyprodukowali czytniki laserowe, wyposażone w inteligentne terminale. Natomiast wydaje się, że produkcja prostych czytników, opartych o układ: dioda świecąca — fototranzystor (wszystko w jednej obudowie) jest w zasięgu możliwości naszego przemysłu. A może jakaś spółdzielnia zacznie robić takie czytniki? Wobec perspektywy masowego stosowania mikroprocesorów, czytniki do odczytu kodu kreskowego mają szansę stać się oczyma tych maszyn.

Pomóżmy mikroprocesorom opanować i tę dziedzinę naszego życia.

JAKUB TATARKIEWICZ

Rastrowe systemy graficzne (przedstawione serią artykułów w ubiegłorocznych numerach INFORMATYKI — 3, 5—9) umożliwiają dwa różne zastosowania komputerów. W pierwszym — nazywanym grafiką komputerową — używane są one m.in. do projektowania statków, samolotów, budynków czy obwodów scalonych; służą też do tworzenia różnorodnych form graficznych (mapy, diagramy), filmów animowanych itp. Cechą charakterystyczną tego zastosowania jest to, że obrazy przedstawione na ekranie monitora są w istocie tworzone przez użytkownika systemu.

Równie często rastrowe systemy graficzne używane są w komputerowym przetwarzaniu obrazów. W tym przypadku, obraz jako taki już istnieje, a zadaniem systemu komputerowego jest przedstawienie go w formie umożliwiającej człowiekowi interpretację treści obrazu. Typowe dane wejściowe w tych zastosowaniach stanowią: zdjęcie niewyraźnych odcisków palców, zdjęcie rentgenowskie ze słabo zaznaczonymi granicami poszczególnych struktur, obraz z kamery (telewizyjnej, termowizyjnej, ultradźwiękowej) z zakłóceniami, nieostre zdjęcie mikroskopowe, małokontrastowe zdjęcie satelitarne czy lotnicze.

Cele i problemy obu tych zastosowań, używane w nich metody i przyjmowane rozwiązania są zdecydowanie różne, niemniej w obu przypadkach najlepsze rezultaty można uzyskać za pomocą systemów, które umożliwiają bezpośrednie współdziałanie użytkownika w tworzeniu lub przetwarzaniu obrazów. (Red.)

JACEK OWCZARCZYK
Warszawa

Interakcyjny system przetwarzania obrazów

Istnieją pewne prawidłowości w wykorzystywaniu komputerów w różnych dziedzinach działalności człowieka i komputerowe przetwarzanie obrazów z pewnością nie jest tu wyjątkiem. Po pierwsze — początkowe zastosowania były związane ze zbrojeniami. Po drugie — zamówienia wojskowe pozwoliły na podjęcie badań na taką skalę, że w krótkim czasie malejące koszty produkcji niezbędnego sprzętu umożliwiły zastosowania cywilne. Te ostatnie są już zresztą bardzo szerokie — wydać by się mogło nawet, że przeznaczone dlań systemy tworzone były jedynie na potrzeby geologii, rolnictwa, medycyny czy przemysłu.

Przetwarzanie obrazów za pomocą systemów komputerowych służy zazwyczaj rozwiązaniu bardziej złożonych problemów, a jego głównym celem jest uzyskanie takiej formy obrazu, która umożliwi człowiekowi łatwiejszą interpretację zawartych w nim informacji. Różnorodność metod cyfrowego przetwarzania obrazów, a także różnorodność jego celów — powodują, że w procesie uzyskiwania wymaganej formy obrazu konieczne jest bezpośrednie współdziałanie człowieka z systemem komputerowym. To stwierdzenie leży u podstaw interakcyjnych systemów przetwarzania obrazów.

Podanie ścisłej definicji systemu interakcyjnego nie wydaje się możliwe. W niniejszym arytkule będzie on rozumiany jako system, w którym — dla rozwiązania określonego problemu — zostały stworzone warunki optymalnego wykorzystania możliwości zarówno użytkownika, jak i systemu komputerowego.

Produkowane obecnie systemy przetwarzania obrazów oparte są najczęściej na konfiguracjach dwukomputerowych. Wśród przyczyn, które wykształciły taką strukturę systemu, można wymienić, oprócz względów ekonomicznych, zwiększenie efektywności procesu przetwarzania. Czynnikiem warunkującym realizację tej struktury był postęp w technologii (mikroprocesory) i organizacji (rozłożona inteligencja) systemów komputerowych.

Sposób realizacji procesów obliczeniowych w interakcyjnym systemie przetwarzania obrazów zależy zarówno od mocy obliczeniowej użytego w systemie graficznym systemu (mikro)komputerowego, jak również od mocy współpracującego systemu komputerowego. W niniejszym artykule skoncentrowano się na rozwiązaniach, które mogą być zrealizowane stosunkowo niewielkim kosztem, preferując rozwiązania sprzętowe. Uzyskanie wymaganych parametrów systemu drogą programową jest znacznie trudniejsze. Z oczywistych względów w rozważaniach pominięto możliwość wykorzystywania w roli systemu towarzyszącego komputerów typu CRAY-1 czy CYBER-205.

UWARUNKOWANIA EFEKTYWNEJ WSPOŁPRACY

Z punktu widzenia organizacji systemów komputerowych — interakcyjny system przetwarzania obrazów może być traktowany tak jak system o rozłożonej inteligencji, w którym — dla osiągnięcia lepszej jego efektywności — jednym z ogniw jest użytkownik. Takie ujęcie systemu interakcyjnego pociąga za sobą konieczność uwzględnienia w jego konstrukcji charakterystyki użytkownika (w sensie jego zdolności percepcji i interpretacji). Wprowadzenie „modelu" użytkownika jest niezbędne m.in. przy ustalaniu podziału zadań i przy określaniu sposobów wymiany informacji pomiędzy użytkownikiem i pozostałymi elementami systemu.

W systemach człowiek-komputer szczególnie użyteczne są takie zdolności użytkownika, jak: inteligentne reagowanie w sytuacjach nietypowych, umiejętność podejmowania decyzji czy wprawa w przezwyciężaniu dwuznaczności w strukturach językowych [14]. Dodatkowo wymienić można zdolność uczenia się i wykorzystywania nabytego doświadczenia oraz — co jest szczególnie ważne — zdolność całościowego postrzegania i dokonywania wstępnej segmentacji, tzn. wyodrębniania obszarów obrazu, zawierających potencjalnie najwięcej informacji. Usuwanie nadmiaru informacji jest bowiem pierwszym etapem analizy obrazów.

Efektywna wymiana informacji pomiędzy użytkownikiem i systemem komputerowym w trakcie przetwarzania wymaga prostego i niejako naturalnego sposobu identyfikowania poszczególnych elementów obrazu. Przetwarzanie obrazów „fragmentami" ma kilka aspektów. Po pierwsze — taki tryb przetwarzania zwiększa znacznie szybkość ob-

liczeń, w wielu przypadkach interesująca użytkownika informacja zawarta jest jedynie w małej części obrazu [27], a więc nie ma sensu przetwarzanie „nieinteresującego tła". Po drugie — niektóre operacje (np. zwiększanie kontrastu, stosowane automatycznie do całych obrazów) nie zawsze pozwolą na wydobycie interesujących szczegółów w poszczególnych fragmentach — dotyczy to głównie obrazów z dużą liczbą regionów [24]. Wydaje się również, że taki tryb przetwarzania zawiera pewne aspekty ergonomiczne — umożliwia on użytkownikowi ocenę skuteczności obranej przez niego metody (lub jej parametrów) na małym fragmencie, zanim zostanie przetworzony cały obraz. W niektórych sytuacjach poprawia on także percepcję wzrokową, np. przy rozpoznawaniu szczegółów, gdy wybrany fragment jest już ostry, natomiast pozostała część obrazu jest w dalszym ciągu nieczytelna.

Ważnym czynnikiem, który powinien być uwzględniony w takim systemie interakcyjnym jest szczególny, noszący znamiona eksperymentu, charakter przetwarzania danych. Wydobycie żądanej informacji odbywa się tu często na zasadzie prób, wielokrotnego sprawdzania skuteczności różnych metod. Oznacza to, że organizacja obliczeń powinna zawierać rozwiązania pozwalające niewielkim kosztem odtworzyć poprzednią wersję obrazu, jeżeli zaproponowana przez użytkownika metoda nie przyniosła spodziewanych rezultatów.

W pracy interakcyjnej istotne znaczenie ma czas odpowiedzi systemu komputerowego. Krytyczna (maksymalna) wartość czasu odpowiedzi to dwie sekundy [16]. Przekroczenie jej powoduje gwałtowny spadek wydajności umysłu użytkownika. Dłuższy czas odpowiedzi może wystąpić jedynie w przypadku zakończenia pewnego etapu analizy danych, przy czym ów etap jest każdorazowo określony przez użytkownika [16].

Czas odpowiedzi systemu komputerowego, niezbędny dla zapewnienia warunków pracy interakcyjnej, pozwala zatem oszacować wymaganą szybkość obliczeń. Jeżeli założymy, że w ciągu dwóch sekund powinien być przetworzony fragment obrazu zawierający 16 K elementarnych punktów (jest to typowa wielkość obrazu używana w ocenie sprawności algorytmów cyfrowego przetwarzania obrazów [4, 32], to wymagana szybkość obliczeń jest rzędu 120 μs/punkt.

Niezbędnym elementem jest tu oczywiście rastrowy system graficzny [18, 20, 21] umożliwiający użytkownikowi obserwację i ocenę wyników obliczeń (zmian przetwarzanych obrazów). Pamięć obrazowa systemu graficznego pełni szczególną — z punktu widzenia organizacji obliczeń — rolę. Pożądaną jej właściwością jest możliwość zmieniania w trakcie obliczeń jej organizacji (tzn. wielkości pojedynczej płaszczyzny i liczby płaszczyzn). W praktyce oznacza to, że przy ustalonej pojemności pamięci obrazowej, wielkość i liczba przechowywanych w niej obrazów jest zmienna. Przykładowo — mogą być przechowywane jednocześnie obraz wejściowy i wynikowy.

Możliwość przetwarzania obrazu w wybranych przez użytkownika fragmentach wymaga przeznaczenia jednej z płaszczyzn pamięci obrazowej na cyfrową reprezentację analizowanego obszaru. Płaszczyzna ta, nazywana w dalszej części artykułu płaszczyzną AR, pełni funkcję rejestru maskującego, a ustalona wartość w poszczególnych jej bitach wskazuje te elementy obrazu, które należą do badanego regionu. Niejako naturalnym dla użytkownika sposobem wskazywania wybranego obszaru na ekranie monitora jest identyfikacja jedynie niektórych punktów brzegowych — za pomocą znacznika sterowanego graficznym urządzeniem wejściowym [21]. Prawidłowe wypełnienie płaszczyzny AR powinno być funkcją odpowiedniego programu.

Ważną rolę w omawianych sytemach, oprócz psychologicznych uwarunkowań dialogu człowiek-system komputerowy [5, 14, 25, 30], odgrywają zagadnienia percepcji wzrokowej [22]. Znajomość i uwzględnianie w konstrukcji systemu tych zagadnień pozwala nie tylko zwiększyć skuteczność procesu przetwarzania obrazów, lecz również uzyskać mniejszym kosztem wymagane parametry systemu. Czas odpowiedzi systemu — na przykład — związany jest z dokładnością przeprowadzanych obliczeń. Jeżeli ich wyniki są analizowane przez użytkownika jedynie wzrokowo, to dokładność obliczeń nie musi być duża, gdyż — z uwagi na ograniczone możliwości percepcyjne (przeciętny użytkownik może rozróżnić na ekranie monitora jedynie ok. 50 poziomów szarości) — poszczególne punkty obrazu i tak nie będą rozróżniane. Z tego względu można często stosować szybsze algorytmy obliczeń [15].

Przykładem znaczenia znajomości zagadnień percepcji wzrokowej, a jednocześnie pokazującym złożoność problemu, jest także sposób wyświetlania obrazów w trakcie ich przetwarzania. Cechą pożądaną jest utrzymywanie zbliżonego poziomu luminancji kolejnych obrazów, tak aby uniknąć konieczności adaptacji sytemu wzrokowego do różnych warunków oświetlenia (czas adaptacji oka, pozwalający osiągnąć taką samą rozdzielczość, może wynosić nawet kilkadziesiąt minut). Z drugiej strony — rozdzielczość oka związana jest z kontrastowością wyświetlanego obrazu, a więc wynik operacji, np. detekcji brzegów, powinien być wyświetlony w formie czarnych linii na białym tle, co jednak powoduje, że dla typowej liczby brzegów wyświetlany obraz jest bardzo jasny.

Należy podkreślić, że optymalne warunki współpracy są uzyskiwane przez dostosowywanie systemu komputerowego do możliwości, zdolności percepcyjnych, a nawet upodobań użytkownika systemu.

KLASYFIKACJA PROCESÓW OBLICZENIOWYCH

Czynnikiem warunkującym stosowanie techniki cyfrowej w przetwarzaniu obrazów (czy innych wiadomości nieznakowych) jest ich uprzednia dyskretyzacja. Proces ten jest obecnie coraz częściej wykonywany już w momencie pozyskiwania danych, gdyż umożliwia on stosowanie techniki cyfrowej również do rejestracji i transmisji obrazów. Przykładem mogą tu być obrazy satelitarne, które są rejestrowane i przesyłane do naziemnych stacji odbiorczych w postaci dyskretnej.

Bez względu jednak na to, czy proces dyskretyzacji przebiega już w momencie pozyskiwania danych, czy też dyskretyzowana jest utrwalona kopia obrazu (np. zdjęcie fotograficzne), zasada tego procesu pozostaje taka sama: zawiera on dwa następujące etapy:

- **rastrowanie** — podział obrazu na elementach pola (punkty) i przyporządkowanie każdemu punktowi pewnej liczby — najczęściej jest to średnia wartość rejestrowanej wielkości fizycznej (dla elementarnego pola)
- **kwantowanie** — przekształcenie otrzymanego w powyższy sposób zbioru liczb w zbiór dyskretny (przeliczalny).

Uzyskana dwuwymiarowa tablica liczb nazywana będzie obrazem cyfrowym, a jej elementy — elementami lub punktami obrazu.

Cechą charakterystyczną obrazów cyfrowych jest stosunkowo niewielka liczba bitów wymagana do kodowania poszczególnych elementów (najczęściej wystarcza osiem bitów) i jednocześnie stosunkowo duże wymiary tablic, np. obraz telewizyjny (cyfrowy) wymaga tablicy o wymiarach 767 × 575, a wspomniane wcześniej obrazy satelitarne to tablice o wymiarach 3240 × 2340 [33].

Przetwarzanie obrazów może być traktowane jako wykonywanie pewnych funkcji na obrazach cyfrowych. Z punktu widzenia organizacji procesów obliczenowych, wygodnie jest podzielić typowe operacje cyfrowego przetwarzania obrazów na trzy podstawowe grupy: operacje jednoelementowe, operacje na otoczeniach i transformacje obrazów. Podstawą tej klasyfikacji jest liczba elementów obrazu wejściowego używanych w obliczeniach pojedynczego elementu obrazu wyjściowego. Jak zostanie pokazane w dalszej części artykułu, realizacja operacji każdej z grup wymaga innych zasobów obliczeniowych.

W operacjach jednoelementowych wartość obliczana w dowolnym punkcie obrazu wyjściowego zależy jedynie od wartości z odpowiadającego (o tych samych współrzędnych) punktu obrazu wejściowego. Przykładem mogą tu być operacje nazywane ogólnie modyfikacją histogramu [26], a służące m.in. do zwiększania kontrastu obrazu [8, 12, 17]. Przykładem jest też kwantowanie dwustopniowe, a także wiele innych przekształceń (liniowych i nieliniowych) elementów obrazu wejściowego [29].

W operacjach na otoczeniach, uważanych za najbardziej powszechną klasę obliczeń cyfrowego przetwarzania obrazów [3, 4], wartość w dowolnym punkcie obrazu wyjściowego zależy od wartości z odpowiadającego punktu obrazu wejściowego i od wartości z otoczenia tego punktu. Zwykle używane są tu, zilustrowane w dalszej części artykułu

dwa typy otoczenia: czteroelementowe i ośmioelementowe [26]. Jako przykłady operacji na otoczeniach można wymienić: zwiększanie ostrości obrazu (operator Laplace'a), wygładzanie (ang. smoothing), detekcja brzegów, filtrowanie.

W transformacjach obrazów wszystkie elementy obrazu wejściowego, traktowanego jako macierz wartości, używane są w obliczeniach poszczególnych elementów obrazu wyjściowego. Przykładami tej klasy obliczeń może być transformacja Fouriera (analiza spektralna obrazu), transformacja Hadamarda, Karhunen-Loéve'a i inne.

PROCESOR OPERACJI JEDNOELEMENTOWYCH

W praktycznych zastosowaniach do kodowania poszczególnych elementów obrazów cyfrowych wymagana jest stosunkowo niewielka liczba różnych wartości (ograniczona także zdolnościami percepcyjnymi użytkownika do rozpoznawania ograniczonej liczby poziomów szarości czy barw [22]), a tym samym — niewielka liczba argumentów przekształceń jednoelementowych. Pozwala to na względnie prostą realizację tych operacji za pomocą dodatkowej, odpowiednio wypełnionej tablicy. Argument przekształcenia jest przy tym adresem elementów tej tablicy, a zapisane tam wartości — żądanym wynikiem przekształcenia. I chociaż powyższa zasada tablicy pośredniej może być wykorzystywana bez względu na sposób implementacji procesora operacji jednoelementowych, to w praktycznych rozwiązaniach — z uwagi na zalety i prostotę realizacji sprzętowej — realizacja programowa nie jest stosowana.

Tablica pośrednia, stanowiąca dodatkowy blok pamięci, włączony pomiędzy pamięcią obrazową i układem formowania sygnału wizyjnego, umożliwia dokonywanie modyfikacji wyświetlanego obrazu (za pomocą operacji jednoelementowych) bez zmian zawartości pamięci obrazowej.

Czas wykonywania operacji jednoelementowych zależy od rodzaju przekształcenia. Operacje jednoelementowe, które obejmują pewne stałe (niezależne od treści przetwarzanego obrazu) przekształcenia, np. logarytmowane wartości z pamięci obrazowej, są wykonywane przez przepisanie do tablicy pośredniej pewnego stałego bloku danych, a więc czas realizacji pokrywa się z czasem aktualizacji tablicy pośredniej. Biorąc pod uwagę jej niewielkie rozmiary, aktualizacja może być przeprowadzona w ciągu kilku milisekund (najlepiej — w trakcie pionowego powrotu plamki w rastrowym systemie graficznym).

Operacje jednoelementowe, w których wartości elementów tablicy pośredniej są zależne od treści przetwarzanego obrazu i muszą być obliczone, wymagają zazwyczaj wyznaczenia histogramu częstości poszczególnych wartości z analizowanego obszaru. Dlatego czas realizacji tych operacji zależy w dużej mierze od szybkości wyznaczania histogramu. Przedstawiony w dalszej części artykułu sposób sprzętowej aktualizacji rejestrów adresujących pamięci obrazowej pozwala na obliczanie histogramu dla 16 K elementów w ciągu kilkudziesięciu milisekund, a więc umożliwia uzyskanie wymaganego czasu odpowiedzi również dla tej grupy operacji jednoelementowych.

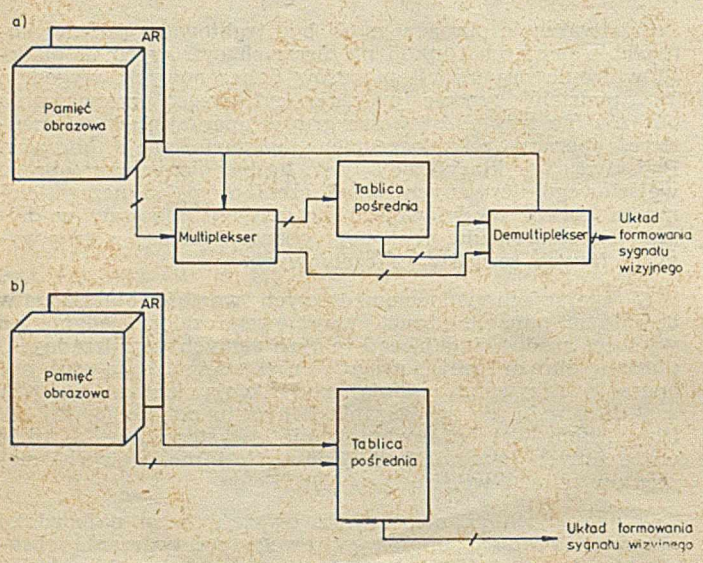
Wykonywanie operacji jednoelementowych jedynie na punktach z zadanego obszaru może być realizowane bądź przez stosowanie układów multipleksera i demultipleksera (rys. 1a), bądź przez traktowanie bitu z płaszczyzny AR jako dodatkowego bitu w szynie adresowej tablicy pośredniej i tożsamościowe wypełnienie jej elementów dla wartości spoza analizowanego obszaru (rys. 1b).

Zastosowania układu tablicy pośredniej, omówionego tu pod kątem wykonywania operacji jednoelementowych, są w systemach przetwarzania obrazów znacznie szersze [29]. W systemach wielokanałowych, w których każdy kanał ma odrębną pamięć obrazową i układ tablicy pośredniej, układ taki stwarza nowe możliwości realizacji niektórych procesów obliczeniowych [2].

PROCESOR OPERACJI NA OTOCZENIACH

Realizacja operacji na otoczeniach za pomocą procesorów macierzowych jest wprawdzie wielokrotnie szybsza niż przy użyciu konwencjonalnych systemów komputerowych [3, 4], niemniej z uwagi na ciągle wysokie koszty sprzętu, zakres stosowania takich procesorów jest ograniczony do stosunkowo wąskiej klasy systemów. Realizacja tych operacji za pomocą typowych systemów mikrokomputerowych jest rozwiązaniem tańszym, niemniej uzyskanie tu szybkości obliczeń wymaganej w interakcyjnym systemie przetwarzania obrazów wymaga dodatkowych układów i specjalnej organizacji obliczeń. Przedstawiony poniżej sprzęt jest przykładem takiego rozwiązania — umożliwia on wykonywanie operacji na otoczeniach z wymaganą szybkością, przy względnie niedużych kosztach.

W typowych operacjach na otoczeniach liczba obliczeń jest stosunkowo niewielka [4], natomiast czasochłonne jest przesyłanie danych z pamięci obrazowej — z uwagi na konieczność ustawiania rejestrów X, Y przed przesyłaniem kolejnych danych z otoczenia analizowanego punktu. Załóżmy, że rejestry adresujące X, Y są skonstruowane z liczników rewersyjnych i mogą być modyfikowane za pomocą czterech linii X+, X—, Y+, Y—. Załóżmy także, że liczba płaszczyzn pamięci obrazowej pozwala na przechowywanie jednocześnie: dwóch obrazów cyfrowych, płaszczyzny AR i dodatkowej płaszczyzny roboczej. Dodatkowy rejestr AMR (Addressing Mode Register), którego zadaniem jest modyfikacja rejestrów X, Y po każdorazowym dostępie do pamięci obrazowej, określa dwa typy adresowania: proste i sekwencyjne. Możliwe sposoby adresowania są przedstawione w tabeli.



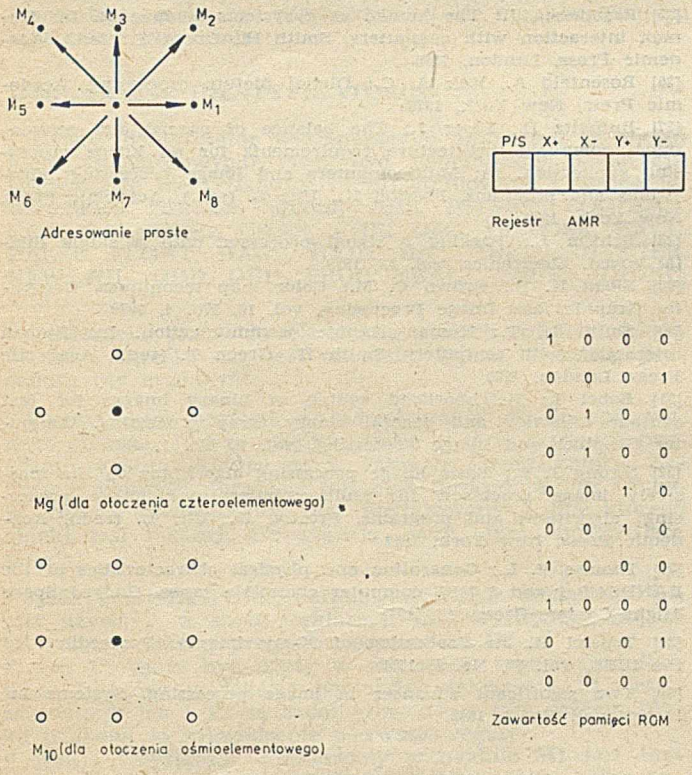
Rys. 1. Wybór elementów z analizowanego regionu za pomocą układów multipleksera i demultipleksera (a) lub przy użyciu powiększonej tablicy pośredniej (b) w realizacji operacji jednoelementowych

Sposoby adresowania

Adresowanie	Sposób	Zawartość rejestru AMR				
		P/S	X+	X—	Y+	Y—
Proste	M_0	0	0	0	0	0
	M_1	0	1	0	0	0
	M_2	0	1	0	0	1
	M_3	0	0	0	0	1
	M_4	0	0	1	0	1
	M_5	0	0	1	0	0
	M_6	0	0	1	1	0
	M_7	0	0	0	1	0
	M_8	0	1	0	1	0
Sekwencyjne	M_9	1	1	1	0	0
	M_{10}	1	0	0	1	1
	M_{11}	1	1	1	1	1

Adresowanie proste M_1—M_8 (rys. 2) jest używane w większości systemów przetwarzania obrazów i umożliwia przesyłanie danych do pamięci obrazowej na osiem różnych sposobów („w ośmiu kierunkach"). Tryby adresowania sekwencyjnego M_9, M_{10} używane są dla dwóch najczęściej stosowanych typów otoczenia: czteroelementowego i ośmio-

elementowego (rys. 2). Tryb adresowania sekwencyjnego M_{11} pozwala na przesłanie do systemu mikrokomputerowego wartości wskazanych przez płaszczyznę AR. Jest on używany m.in. przy obliczaniu histogramu częstości poszczególnych wartości z analizowanego regionu. Ciąg modyfikacji rejestrów X, Y, generowany podczas adresowania sekwencyjnego jest przechowywany w pamięci ROM (rys. 2), a rejestry X, Y są sterowane bądź bezpośrednio z rejestru AMR, bądź z pamięci ROM (bit PS w rejestrze AMR steruje przełączaniem).



Rys. 2. Sposoby adresowania elementów obrazu

W trakcie adresowania sekwencyjnego, z płaszczyzną roboczą, której początkowa zawartość jest identyczna z płaszczyzną AR, związany jest rejestr przesuwający (ang. serial in/parallel out). Zawartość tego rejestru jest uzupełniana podczas każdego (z wyjątkiem pierwszego) odczytu danych z pamięci obrazowej, a więc po wykonaniu ciągu modyfikacji zawiera on kod otoczenia. Kod ten jest używany do określenia elementu z otoczenia, który będzie przetwarzany w następnej kolejności. Operacja znajdowania adresu nowego elementu jest wykonywana zawsze wtedy, gdy aktualną modyfikacją z pamięci ROM są „same zera". Równocześnie bit płaszczyzny roboczej, odpowiadający elementowi, dla którego zostały przeprowadzone obliczenia, jest zerowany.

Wybór kolejnych elementów obrazu należących do analizowanego regionu (zaznaczonych w płaszczyźnie AR) może być realizowany układowo za pomocą tablic pośrednich na podstawie kodu otoczenia (przy ustalonym kierunku przeszukiwania) [31]. Wykonanie żądanej operacji dla wszystkich punktów sygnalizowane jest przerwaniem.

Podczas adresowania sekwencyjnego, z każdą modyfikacją (z wyjątkiem ostatniej) związany jest odczyt danych z pamięci obrazowej natomiast z ostatnią — zapis danych do tej pamięci (przesłanie wyniku operacji na otoczeniach). W celu dalszego zwiększenia szybkości obliczeń, ustalona grupa bitów (np. mniej znaczący bajt) rejestr DATA, służącego do komunikacji z pamięcią obrazową, może być dołączona sprzętowo podczas operacji odczytu do płaszczyzn pamięci obrazowej zawierających obraz wejściowy, natomiast podczas operacji zapisu — do płaszczyzn zawierających obraz wynikowy. W ten sposób wynik obliczeń nie musi być przesuwany programowo we właściwe miejsce rejestru DATA.

Powyższe rozwiązanie pozwala nie tylko uzyskać wymaganą szybkość obliczeń dla operacji na otoczeniach za pomocą typowego systemu mikrokomputerowego, lecz również powoduje, że struktura oprogramowania dla tych operacji jest bardzo prosta. Każdy z programów stanowi pętlę i po zainicjowaniu jest on wykonywany aż do momentu przerwania. W programie zawarty jest jedynie fragment realizujący czytanie z rejstru DATA, algorytm danej operacji na otoczeniach oraz zapis do rejestru DATA: nie zawiera on natomiast aktualizacji rejstrów X, Y, buforowania czy przesuwania wyniku operacji w rejestrze DATA.

PROCES TRANSFORMACJI OBRAZÓW

Możliwości implementacji procesora transformacji obrazów zostaną przedstawione na przykładzie transformacji Fouriera. Transformacja ta została wybrana nie tylko z uwagi na największe znaczenie w wielu zastosowaniach [34], lecz również ze względu na możliwość ilustracji różnorodnych sposobów jej realizacji.

Realizacja programowa (za pomocą towarzyszącego systemu komputerowego) jest z reguły rozwiązaniem najtańszym, niemniej może być dosyć czasochłonna. Wymaganą szybkość obliczeń można uzyskać bądź za pomocą procesorów macierzowych (szybka transformacja Fouriera dla obrazu cyfrowego o wymiarach 512 × 512 wykonywana jest w ciągu 25 sekund [9]), bądź za pomocą oprogramowania układowego (czas obliczeń dla obrazu o wymiarach 128 × ×128 wynosi dwie sekundy [35]). Inne specjalizowane urządzenia, których konstrukcja jest względnie prosta i tania, dzięki specjalnie zaprojektowanym do tego celu układom scalonym, umożliwiają wykonywanie szybkiej transformacji Fouriera (dla 1024 punktów) w czasie rzeczywistym dla kanału danych o szerokości pasma 50 kHz [28].

Koszt procesora macierzowego jest znacznie większy niż koszt specjalizowanego układu, niemniej jego zastosowania w interakcyjnym systemie przetwarzania obrazów mogą być znacznie szersze [6, 7] i nie ograniczają się do realizacji wąskiej klasy algorytmów. Mówiąc o kosztach, należy podkreślić, że dostosowywanie systemu do wymagań użytkownika nie może powodować zbyt dużych nakładów, gdyż wyznaczają one zakres stosowania i liczbę użytkowników systemu.

Wydaje się, że w przypadku transformacji Fouriera, przyjmowane dotąd wymagania odnośnie szybkości obliczeń i czasu odpowiedzi systemu, mogą być nieco złagodzone, gdyż wykonanie tej operacji stanowi pewien zamknięty etap w analizie treści obrazu.

* * *

Przedstawione w niniejszym artykule problemy obliczeń w interakcyjnym systemie przetwarzania obrazów dotyczą operacji charakterystycznych dla tego typu zastosowań. Problemów jest w istocie więcej, a dokonywanie jakiegokolwiek wyboru może być jedynie umowne, gdyż w praktycznych zastosowaniach przetwarzania obrazów stanowi trudną do wydzielenia część bardziej złożonego procesu obliczeniowego. W artykule skoncentrowano się na rozwiązaniach układowych i dlatego problemy konstrukcji oprogramowania systemów przetwarzania obrazów [15, 23] zostały pominięte. Omówione zostały jedynie najważniejsze elementy interakcyjnego systemu przetwarzania obrazów. Konfiguracje spotykane w praktyce są zazwyczaj bardziej rozbudowane i wyposażone w szereg specjalizowanych układów [19], a czasami bardzo potężne konfiguracje wielomaszynowe także określane są tym terminem [1].

Efektywna współpraca człowieka z sytemami komputerowymi obejmuje wiele problemów; ich liczba rośnie w miarę jak użytkownikami tych systemów stają się (tak jak w przypadku przetwarzania obrazów) ludzie nie będący zawodowymi informatorami. Stworzenie im warunków **rzeczywistego** współdziałnia zapoczątkuje nowy etap zastosowań komputerów.

LITERATURA

[1] Bracken P. A., Dalton J. T., Quann J. J.: AOIPS — an interactive image processing system. National Computer Conference Proceedings, 1978

[2] Clark D. R.: The technical foundations of computer image-making. In: Computers for imagemaking, Clark D. R. (red.), Pergamon Press, Oxford, 1981
[3] Cordella L. P., Duff M. J., Levialdi S.: Thresholding: a challenge for parallel processing. Computer Graphics and Image Processing, vol. 6, No. 2, 1977
[4] Cordella L. P., Duff M. J., Levialdi S.: An analysis of computational cost in image processing: a case study. IEEE Transactions on Computers, vol. C-27, No. 10, 1978
[5] Dean M.: How a computer should talk to people. IBM Systems Journal, vol. 21, No. 4, 1982
[6] Duff M. J.: Parallel processing techniques. In: Pattern Recognition, Batchelor B. G. (red.), Plenum Press, New York, 1978
[7] Duff M. J.: Parallel algorithms and their influence on the specification of application problems. In: Multicomputers and image processing: algorithms and programs, Preston K., Uhr L. (red.), Academic Press, New York, 1982
[8] Frei W.: Image enhancement by histogram hyperbolization. Computer Graphics and Image Processing, vol. 6, No. 3, 1977
[9] Gnauck G. E., Hu P. C.: Image processing and geographic information systems and their application to resource management and exploration. ESL Inc., Sunnyvale, 1981
[10] Halé J. A., Saraga P.: Digital image processing. In: Pattern Recognition, Batchelor B. G. (red.), Plenum Press, New York, 1978
[11] Haralick R. M., Minden G.: KANDIDATS: an interactive image processing system. Computer Graphics and Image Processing, vol. 8, No. 1, 1978
[12] Hummel R.: Image enhancement by histogram transformation. Computer Graphics and Image Processing, vol. No. 2, 1977
[13] Lundgren K., Granlund G. H.: Image processing applications enabled by MIMD processor structure. In: Multicomputers and image processing: algorithms and programs, Preston K., Uhr L. (red.). Academic Press, New York, 1982
[14] Martin J.: Design of man-computer dialogues. Prentice-Hall, Englewood Cliffs, 1973
[15] Michalski M., Owczarczyk J.: Oprogramowanie systemów przetwarzania obrazów. Technika Poszukiwań Geologicznych, nr 1, 1984
[16] Miller R. B.: Response time in man-computer transactions. AFIPS Conference Proceedings, Thompson Book Company, Washington, 1968
[17] Owczarczyk J.: Cyfrowe metody zwiększania kontrastu obrazów satelitarnych. Technika Poszukiwań Geologicznych, nr 2, 1983
[18] Owczarczyk J.: Rastrowe systemy graficzne. Informatyka, nr 3, 1983
[19] Owczarczyk J.: Parametry użytkowe wybranych rastrowych systemów graficznych. Informatyka, nr 5, 1983
[20] Owczarczyk J.: Projektowanie systemu graficznego. Część 1. Informatyka, nr 6, 1983
[21] Owczarczyk J.: Projektowanie systemu graficznego. Część 2. Informatyka, nr 7/8, 1983
[22] Owczarczyk J.: Systemy graficzne: percepcja wzrokowa. Informatyka, nr 9, 1983
[23] Owczarczyk J., Michalski M.: Organizacja zbiorów i procedury dostępu w komputerowych systemach przetwarzania obrazów. Technika Poszukiwań Geologicznych, nr 6, 1983
[24] Pal S. K., King R. A.: Histogram equalization with S and II functions in detecting X-ray edges. Eectronics Letters, vol. 17, No. 9, 1981
[25] Rasmussen J.: The human as a systems component. In: Human interaction with computers, Smith H., Green T. (red.), Academic Press, London, 1980
[26] Rosenfeld A., Kak A. C.: Digital picture processing. Academic Press, New York, 1976
[27] Rutovitz D., Piper J.: The balance of special and conventional computer architecture requirements for an image processing application. In: Multicomputers and image processing: algorithms and programs, Preston K., Uhr L. (red.), Academic Press, New York, 1982
[28] Schirm L.: Packing a signal processor onto a single digital board. Electronics, vol. 52, 1979
[29] Sloan K. R., Brown C. M.: Color map techniques. Computer Graphics and Image Processing, vol. 10, No. 4, 1979
[30] Smith H. T.: Human-computer communication. In: Human interaction with computers, Smith H., Green T. (red.), Academic Press, London, 1980
[31] Sobel I.: Neighborhood coding of binary images for fast contour following and general binary array processing. Computer Graphics and Image Processing, vol. 8, No. 1, 1978
[32] Strong J. P.: Basic image processing algorithms on the massively image processor. In: Multicomputers and image processing: algorithms and programs, Preston K., Uhr L. (red.), Academic Press, New York, 1982
[33] Thomas V. L.: Generation and physical characteristics of the LANDSAT 1 and 2 MSS computer comatible tapes. Godard Space Flight Center, Greenbelt, 1975
[34] Trussel H. J.: Processing of X-ray images. Proceedings of the IEEE, vol. 69, No. 5, 1981
[35] Two significant advances in image processing. Systems International, No. 2, 1982.

PIOTR FUGLEWICZ
CNPSS MERASTER
Katowice

MODULA-2 — język lat osiemdziesiątych (3)

Opis składni

Prostota składni języka MODULA-2 skłania do przedstawienia możliwie pełnego jej opisu w ramach jednego artykułu, jednakże ograniczenia objętości nie pozwalają na prezentację opisu formalnego i szczegółowego. Artykuł oparto w całości na skrypcie N. Wirtha [2]. Część terminologii oraz definicje użytych pojęć przyjęto za książką M. Iglewskiego i in. [1], w takim zakresie, w jakim są wspólne dla PASCALA i MODULI-2.

SŁOWNIK JĘZYKA

Język stanowi niekończony zbiór zdań (programów) napisanych zgodnie z jego składnią. Każde zdanie (program) jest skończonym ciągiem symboli ze skończonego słownika. Słownik MODULI-2 składa się z identyfikatorów, liczb, napisów, operatorów i ograniczników. Obiekty te, nazywane symbolami leksykalnymi, składają się ze znaków.

Identyfikator (nazwa) jest ciągiem liter i cyfr, którego pierwszym znakiem musi być litera.

Liczby dzielą się na całkowite (bez znaku) i rzeczywiste. Zapis liczb całkowitych jest ciągiem cyfr. Jeśli liczba zakończona jest znakiem **B**, jest traktowana jako liczba ósemkowa. Znak **H** oznacza liczbę w zapisie szesnastkowym. Zakończenie liczby znakiem **C** wskazuje reprezentację znaku, którego wartość ósemkowa przedstawia liczba. Taka liczba jest typu CHAR. Liczba całkowita z przedziału **(O, MaxInt)** może być typu INTEGER lub CARDINAL, liczba z przedziału **(MaxInt+1, MaxCard)** jest typu CARDINAL. Dla komputerów szesnastobitowych: **MinInt** = = —**32768, MaxInt** = **32767, MaxCard** = **65535**. Liczba rzeczywista zawsze zawiera kropkę dziesiętną. Opcjonalnie może zawierać mnożnik skalujący. Liczba rzeczywista jest typu REAL.

Napis jest ciągiem znaków ujętych w apostrofy (pojedyncze lub podwójne). Apostrof otwierający i zamykający napis musi być tego samego typu i nie może występować wewnątrz napisu. Napis o długości $n > 1$ jest typu **ARRAY [0..n—1] OF CHAR**. Napis jednoznakowy jest typu CHAR.

Operatory i ograniczniki są specjalnymi znakami, parami znaków lub słowami kluczowymi. Słowa kluczowe składają się tylko z dużych liter i nie mogą być używane w charakterze identyfikatorów: AND, ARRAY, BEGIN, BY, CASE, CONST, DEFINITION, DIV, DO, ELSE, ELSIF, END, EXIT, EXPORT, FOR, FROM, IF, IMPLEMENTATION, IMPORT, IN, LOOP, MOD, MODULE, NOT, OF, OR, POINTER, PROCEDURE, QUALIFIED, RECORD, REPEAT, RETURN, SET, THEN, TO, TYPE, UNTIL, VAR, WHILLE, WITH.

Między dowolnymi dwoma symbolami programu mogą występować komentarze. Komentarz jest ciągiem dowolnych znaków zawartych między parami symboli (* i *). Komentarze mogą być zagnieżdżane i nie mają wpływu na wykonanie programu.

DEKLARACJE IDENTYFIKATORÓW

Każdy identyfikator występujący w programie (z wyjątkiem standardowych) musi być poprzedzony deklaracją. Następujące identyfikatory są predefiniowane i ważne we wszystkich zdaniach języka: ABS, BITSET, BOOLEAN, CAP, CARDINAL, CHAR, CHR, DEC, DISPOSE, EXCL, FALSE, FLOAT, HALT, HIGH, INC, INCL, INTEGER, NEW, NIL, ODD, ORD, PROC, REAL, TRUE, TRUNC, VAL.

Deklaracja służy do określenia stałych właściwości obiektu i określa, czy jest on stałą, typem, zmienną, procedurą czy modułem.

Identyfikator pozwala na odwoływanie się do opisywanego obiektu. Jest to możliwe tylko w tych częściach programu, które znajdują się w zakresie identyfikatora. Zakresem identyfikatora jest ten blok (procedura lub moduł, w którym identyfikator jest deklarowany i dla którego obiekt jest lokalny. Obowiązują przy tym następujące zasady:

- jeśli identyfikator x, definiowany przez deklarację **D1**, jest używany w innej deklaracji (nie wyrażeniu) **D2**, to **D1** musi w tekście programu poprzedzać **D2**
- typ **T1** może być użyty w deklaracji typu wskaźnikowego **I**, który w tekście poprzedza deklarację **T1**, jeżeli zarówno **I**, jak i **T1** są zadeklarowane w tym samym bloku (stanowi to załagodzenie pierwszej reguły)
- jeśli identyfikator definiowany w module **M1** jest eksportowany, jego zakres rozciąga się poza blok, który zawiera **M1** (jeśli **M1** jest jednostką kompilacji, zakres identyfikatora rozciąga się na wszystkie moduły, które importuje **M1**)
- identyfikatory pól deklaracji rekordu są ważne tylko dla desygnatorów pól oraz instrukcji WITH odwołujących się do zmiennej danego typu rekordowego.

Identyfikator może być kwalifikowany. W takim przypadku jest prefiksowany kolejnym identyfikatorem, określającym moduł, w którym jest definiowany identyfikator kwalifikowany. Prefiks i identyfikator rozdzielone są kropką.

Deklaracja stałej przypisuje identyfikatorowi stałą wartość. Identyfikatorowi może być przypisana również wartość wyrażenia stałego. Identyfikator, poprzedzający lewy nawias specyfikacji zbioru, określa typ zbioru. W przypadku gdy identyfikator ten nie występuje, przyjmowany jest standardowy typ BITSET.

Deklaracja zmiennej służy do wprowadzenia zmiennej i przypisania jej niepowtarzalnego identyfikatora typu i struktury. Zmienne, których identyfikatory występują na jednej liście, są jednego typu.

Deklaracje typów

Typ danej określa zbiór wartości, które może przyjmować zmienna, jak również operatory, które mogą być do niej stosowane. Mechanizm definiowania typu umożliwia jednoczesne podanie opisu typu i przyporządkowanie typowi identyfikatora. W przypadku typów strukturalnych (tablice, rekordy i zbiory) deklaracja typu określa również strukturę zmiennych danego typu.

W MODULI-2 istnieje pięć typów podstawowych, które są predefiniowane i opisane standardowymi identyfikatorami: INTEGER, CARDINAL, BOOLEAN, CHAR i REAL.

Typ wyliczeniowy zawiera listę identyfikatorów, które określają tworzące go wartości. Identyfikatory występują w programie jako wartości stałe. Porządek w typie jest zgodny z kolejnością wyliczenia w definicji typu. Numer porządkowy pierwszej wartości wynosi 0.

Typ **T** może być zdefiniowany jako typ okrojony innego podstawowego (z wyjątkiem REAL) lub wyliczeniowego typu **T1**, przez podanie wartości ograniczenia dolnego i górnego. Typ **T1** jest nazywany typem bazowym typ **T** i wszystkie operatory, działające na typie **T1**, działają również na typie **T**. Jeśli ograniczenie dolne jest nieujemną liczbą całkowitą, typem bazowym okrojenia jest typ CARDINAL. Jeśli jest liczba całkowita ujemną, typem bazowym okrojenia jest typ INTEGER.

Typ tablicowy jest strukturą złożoną ze stałej liczby elementów tego samego typu, nazywanego typem składowym. Elementy tablicy są określone przez indeksy. Deklaracja typu tablicowego określa zarówno typy elementów tablicy, jak i typ indeksów. Indeks może być typu wyliczeniowego, okrojonego, BOOLEAN lub CHAR.

Typ rekordowy jest strukturą złożoną ze stałej liczby elementów, które mogą być różnych typów. Deklaracja typu rekordowego określa dla każdego elementu, nazywanego polem, jego typ i identyfikator z nim związany. Zakresem identyfikatora pola jest rekord, w którym identyfikator jest zdefiniowany. Identyfikator ten, użyty wewnątrz desygnatora pola, umożliwia dostęp do zadanego elementu konkretnego rekordu. Typ rekordowy może mieć kilka różnych wariantów. W takim przypadku, pierwsze pole części zmiennej jest polem znacznikowym i jego wartość określa aktualny wariant.

Typ zbiorowy **T1**, definiowany jako **T1 = SET OF T** jest zbiorem wszystkich podzbiorów typu **T**. Typ **T** musi być typem podstawowym (lub okrojeniem) o co najwyżej **WordSize** elementach (dla komputerów szesnastobitowych **WordSize = 16**). Standardowy typ BITSET jest predefiniowany jako **SET OF [0.. WordSize-1]**.

Zmienne typu wskaźnikowego **W = POINTER TO T** przyjmują wartości, wskazujące na zmienne innego typu **T**. Wartość wskaźnika jest określana podczas użycia standardowej procedury alokacji dynamicznej — NEW. Zmienna typu wskaźnikowego może też przyjąć standardową wartość NIL, co oznacza, że obiekt, na który ma wskazywać, nie istnieje.

Zmienna typu procedurowego **T** może przyjmować jako swoją wartość procedurę **P**. Typy parametrów formalnych procedury P muszą być zgodne z typami parametrów procedury **T**. Procedura **P** nie może być zadeklarowana jako lokalna dla innej procedury, jak również nie może być procedurą standardową. Standartowy typ PROC opisuje procedurę bez parametrów **PROC = PROCEDURE**.

Deklaracja procedur

Deklaracja procedury składa się z nagłówka procedury i bloku, będącego treścią procedury. Nagłówek procedury określa identyfikator procedury i jej parametry formalne. Blok zawiera deklaracje i instrukcje. Identyfikator procedury jest powtarzany na końcu deklaracji procedury.

Istnieją dwa rodzaje procedur: procedura właściwa i funkcja. W deklaracji procedur obydwu typów występuje symbol **PROCEDURE**. Funkcja jest wykonywana wskutek wywołania funkcji, (składnik wyrażenia) i zwraca wynik będący argumentem wyrażenia. Deklaracja funkcji wyróżnia się określeniem typu jej wyniku, które występuje po liście parametrów. Treść funkcji musi zawierać instrukcję RETURN. Procedura właściwa jest aktywizowana przez wywołanie procedury.

Wszystkie obiekty zadeklarowane w treści procedury są dla tej procedury lokalne. Wartości zmiennych lokalnych są na zewnątrz nieokreślone. Deklaracje procedury mogą być zagnieżdżone. Każdy obiekt jest deklarowany na określonym poziomie zagnieżdżenia. Obiekty zadeklarowane w module, który tworzy jednostkę kompilacja, są na poziomie 0.

Oprócz parametrów formalnych i obiektów lokalnych, również obiekty zadeklarowane w otoczeniu procedury są znane i dostępne w treści procedury (z wyjątkiem obiek-

tów, które mają taką samą nazwę, jak obiekty zadeklarowane lokalnie).
Możliwe jest stosowanie rekursywnego wywoływania procedury.

Parametry formalne są identyfikatorami. Identyfikatory te są zastępowane przez parametry aktualne podczas wywołania procedury. W MODULI-2 występują parametry przekazywane przez wartość oraz parametry przekazywane przez zmienną. Sposób przekazywania parametru jest określany na liście parametrów formalnych.

Parametry przekazywane przez wartość są podstawiane pod lokalne zmienne procedury, jako wartości początkowe. Parametry przekazywane przez zmienną odpowiadają aktualnej wartości zmiennej, na której działa procedura. Parametry przekazywane przez zmienną są wskazywane przez użycie symbolu **VAR** na liście parametrów formalnych, a przekazywane przez wartość — brakiem tego symbolu.

Parametry formalne są lokalne dla procedury, to znaczy, że ich zakresem jest tekst deklaracji procedury.

Typ każdego parametru jest podany na liście parametrów formalnych. Parametry aktualne przekazywane przez zmienne muszą być zgodne z odpowiadającymi im parametrami formalnymi. Parametry przekazywane przez wartość muszą być zgodne względem podstawienia.

Jeśli parametr jest tablicą, można używać formy **ARRAY OF T** z pominięciem deklaracji części indeksowej. Typ **T** musi być zgodny z typem elementu tablicy, będącej parametrem aktualnym. Zakres indeksów tablicy **N**-elementowej jest odwzorowywany w **{0... N-1}**. Tablica formalna jest dostępna tylko na poziomie elementu lub może występować jako aktualny parametr, którego parametr formalny nie ma specyfikowanej części indeksowej.

Jeśli parametr formalny jest typu procedurowego, to odpowiadający mu parametr aktualny musi być procedurą zadeklarowaną na poziomie **0** lub zmienną (albo parametrem) typu procedurowego takiego rodzaju. Parametr aktualny nie może być procedurą standardową.

W MODULI-2 istnieją następujące (predefiniowane) procedury standardowe:

ABS (x) — wartość bezwzględna
CAP (z) — konwersja liter małych na duże
CHR (x) — znak o numerze **x**, **CHR (x) = VAL (CHAR, x)**
FLOAT (x) — **x** typu CARDINAL przedstawione jako wartość typu REAL
HIGH (t) — górny indeks tablicy **t**
ODD (x) — **x MOD 2 ≠ 0** (nie równa się zero)
ORD (x) — numer kolejny **x** w zbiorze wartości zdefiniowanych przez typ zmiennej
TRUNC (x) — część całkowita liczby rzeczywistej
VAL (T, x) — wartość elementu o numerze kolejnym **x** w typie **T**
DEC (x) — **x : = x—1**
DEC (x, n) — **x : = x—n**
EXCL (z, i) — **z : = z—(i)**
HALT — przerwanie pracy programu
INC (x) — **x : = x+1**
INC (x, n) — **x : = x+n**
INCL (z, i) — **z : = z+(i).**

WYRAŻENIA

Wyrażenie jest konstrukcją określającą zasady obliczania i przypisania zmiennej nowej wartości. Wyrażenie składa się z argumentów i operatorów.

Argumenty

Argumenty, z wyjątkiem literałów, są wskazywane przez desygnatory. Desygnator jest identyfikatorem wskazującym na stałą, zmienną lub procedurę. Identyfikator może być poprzedzony nazwą modułu, może również zawierać selektor, jeśli wskazywany obiekt jest elementem struktury. Jeśli struktura jest tablicą **T**, to desygnator **T [W]** wskazuje na ten element tablicy, którego indeks jest bieżącą wartością **W**.

Typ indeksowy tablicy musi być zgodny z typem wartości wyrażenia **W**. Jeśli struktura jest rekordem, desygnator **R.p** wskazuje pole **p** rekordu **R**. Desygnator **W** wskazuje zmienną, która jest wskazywana przez wskaźnik **W**.

Jeśli wskazywany obiekt jest zmienną, desygnator odnosi się do jego wartości bieżącej. Jeśli obiekt jest funkcją, desygnator bez listy parametrów odnosi się do tej procedury. Jeśli występuje lista parametrów, użycie desygnatora powoduje wywołanie funkcji i podstawienie wartości wyniku. Wymagana jest w takim przypadku zgodność parametrów formalnych i aktualnych.

Operatory

Składnia wyrażenia umożliwia użycie operatorów o czterech klasach priorytetów. Najwyższy priorytet ma operator NOT, niższy operatory (kolejno): multyplikatywne, addytywne i operatory relacji. Ciąg operatorów o tym samym priorytecie jest obliczany od lewej do prawej.

Operatory arytmetyczne +,—,*, dzielenie całkowite — DIV, dzielenie modulo — MOD, z wyjątkiem/ działają na argumentach typu INTEGER, CARDINAL lub typach okrojonych tych typów. Operatory +, —, i * stosuje się również dla argumentów typu REAL, a operator / tylko dla argumentów typu REAL.

Operatory logiczne OR, AND i NOT działają tylko na argumentach typu BOOLEAN i wynik ich działania jest typu BOOLEAN.

Operatory teoriomnogościowe: + (suma zbiorów), — (różnica zbiorów), * (przecięcie zbiorów) i / (różnica symetryczna zbiorów) działają na argumentach typu zbiorowego i wynik ich działania jest tego samego typu.

Operatory relacyjne: =, ≠ (nierówność), <, <= (mniejszość lub równość, zawieranie zbiorów) >, >= (większość lub równość, zawieranie zbiorów), **IN** (zawieranie się, przynależność do zbioru) — dają wynik typu BOOLEAN. Relacja porządkująca określona jest dla typów INTEGER, CARDINAL, BOOLEAN, CHAR i REAL oraz typu wyliczeniowego i typów okrojonych powyższych typów. Relacje = i ≠ stosuje się również dla zbiorów i wskaźników. W wyrażeniu postaci **x IN z** wyrażenie **z** musi być typu **SET OF T**, gdzie **T** jest zgodne z typem **x**.

INSTRUKCJE

Instrukcje dzielą się na proste oraz strukturalne. Instrukcje proste, w przeciwieństwie do strukturalnych, nie zawierają składników, które same są instrukcjami.

Instrukcje proste

Instrukcja podstawienia służy do zamiany bieżącej wartości zmiennej przez nową wartość wyznaczoną wskutek wartościowania wyrażenia. Operator podstawienia ma postać ":=". Desygnator po lewej stronie operatora wskazuje na zmienną. Typ zmiennej musi być zgodny względem podstawienia z typem wyrażenia. Napis o długości **n1** może być podstawiony do zmiennej o długości **n2 > n1**. W takim przypadku zmienna jest dopełniana znakami pustymi **(0C)**.

Wywołanie procedury powoduje realizację treści procedury i może zawierać parametry aktualne, które są podstawiane w miejsce odpowiadających im parametrów formalnych, zdefiniowanych w deklaracji procedury. Odpowiedniość jest ustalana przez kolejność parametrów na obu listach. Parametr przekazywany przez wartość musi być desygnatorem wskazującym zmienną. Jeśli jest to zmienna złożona lub strukturalna, selektor jest wyznaczony w chwili zastąpienia parametru formalnego aktualnym. Parametr przekazywany przez zmienną musi być wyrażeniem. Wyrażenie jest wartościowane przed realizacją procedury i jego wartość jest podstawiana w miejsce parametru formalnego.

Skutkiem działania instrukcji EXIT jest opuszczenie pętli, tzn. przerwanie wykonywania ciągu instrukcji i kontynuacja programu, od instrukcji następnej za terminatorem pętli. Instrukcja opuszczenia pętli jest kontekstowo (ale nie semantycznie) składnikiem instrukcji pętli.

Instrukcja RETURN, po której może następować wyrażenie **W** powoduje zakończenie procedury (lub treści modułu). Wyrażenie W określa wartość zwracaną jako wynik funkcji. W procedurze właściwej, powrót do wywołującego fragmentu programu następuje po napotkaniu terminatora treści procedury, jednak może w niej nastąpić symbol RETURN, określając dodatkowy punkt zakończenia.

Instrukcje złożone

W instrukcji warunkowej wyrażenia występujące po symbolach IF i ELSIF są typu BOOLEAN. Wykonanie instrukcji polega na wartościowaniu tych wyrażeń w kolejności występowania, aż do osiągnięcia wartości TRUE. Wykonywany jest wtedy ciąg instrukcji związany z tym wyrażeniem. Jeśli występuje symbol ELSE, to ciąg instrukcji związany z nim jest wykonywany wtedy i tylko wtedy, gdy wszystkie wyrażenia mają wartość FALSE.

Instrukcja CASE polega na wyznaczeniu selektora i wykonaniu ciągu instrukcji, dla którego wartość etykiety na liście etykiet wyboru odpowiada wyznaczonej wartości. Typ wyrażenia musi być typem prostym (z wyjątkiem REAL), wyliczeniowym lub okrojonym. Typ wszystkich etykiet wyboru musi być zgodny z typem wyrażenia. Żadna wartość nie może wystąpić na liście etykiet wyboru więcej niż raz. Jeśli wartość selektora nie występuje na liście etykiet wyboru, wykonywana jest sekwencja instrukcji występująca po symbolu ELSE.

Instrukcja WHILE służy do wykonywania iteracji w zależności od wartości wyrażenia logicznego. Warunek jest wartościowany przed wykonaniem ciągu instrukcji, zawartym między symbolami DO i END.

Instrukcja REPEAT opisuje iteracje ze sprawdzaniem warunku na końcu. Wykonanie ciągu instrukcji zawartego między symbolami REPEAT i UNTIL powtarza się (co najmniej raz) tak długo, jak długo nie jest spełniony warunek zapisany po symbolu UNITIL.

Instrukcja FOR służy do wielokrotnego wykonania ciągu instrukcji ze zmieniającą się, o stałą wartość, zmienną kontrolną. Zmienna kontrolna nie może być zmienną złożoną lub strukturalną, nie może być importowana, ani nie może być parametrem. Wartość zmiennej kontrolnej nie może być zmieniana przez wewnętrzny ciąg instrukcji. Zmienna kontrolna musi być zgodna względem podstawienia z wartością początkową i końcową pętli. Krok musi być typu INTEGER lub CARDINAL. Jeśli krok nie jest podany, jego wartość domyślna wynosi 1. Wartość zmiennej kontrolnej jest porównywana z wartością końca pętli, przed wykonaniem wewnętrznego ciągu instrukcji.

Instrukcja LOOP służy do wielokrotnego powtórzenia ciągu instrukcji, zawartych między symbolami LOOP i END. Przejście do instrukcji następnej, za terminatorem pętli, następuje po napotkaniu instrukcji EXIT.

Instrukcja WITH umożliwia odwoływanie się do pól zmiennej rekordowej, z pominięciem kwalifikatora. Jeśli desygnator jest zmienną złożoną lub strukturalną, to selektor wyznacza się przed wykonaniem ciągu instrukcji.

MODUŁY

Moduł składa się z deklaracji i ciągu instrukcji. Terminatorami modułu są symbole MODULE i END. Nagłówek modułu zawiera jego nazwę, może też zawierać listy eksportowe i importowe. Lista importowa wymienia identyfikatory, które są zdefiniowane poza modułem, a używane wewnątrz. Lista eksportowa określa, które identyfikatory zdefiniowane wewnątrz modułu mogą być użyte na zewnątrz.

Obiekty lokalne dla modułu mają ten sam zakres co sam moduł. Nie mogą one być lokalne dla procedury zawierającej moduł.

Ciąg instrukcji, który stanowi treść modułu (blok) jest wykonywany podczas wywołania procedury, dla której moduł jest lokalny. Jeśli procedura zawiera kilka modułów, ich treść jest wykonywana w kolejności występowania.

Jeśli identyfikator występuje na liście eksportowej (importowej), wtedy obiekt, którego dotyczy, może być używany wewnątrz (na zewnątrz) modułu, tak jakby był dla modułu lokalny. Jeśli jednak na liście eksportowej po symbolu EXPORT występuje symbol QUALIFIED, wtedy indentyfikatory z tej listy muszą być na zewnątrz prefiksowane identyfikatorem modułu. Taki sposób eksportowania obiektów nazywa się kwalifikowanym. Użycie na liście importowej symbolu FROM umożliwia korzystanie z obiektów eksportowanych w sposób kwalifikowany, na takich zasadach, jakby były eksportowane w normalny sposób.

Jeśli jest eksportowany obiekt typu rekordowego, wszystkie identyfikatory pól są eksportowane, bez konieczności umieszczania ich na liście eksportowej.

UDOGODNIENIA ZALEŻNE SYSTEMOWO

MODULA-2 zawiera pewną liczbę udogodnień służących do programowania operacji na niskim poziomie i programowania procesów. Działanie tych funkcji zależy od konkretnego sprzętu lub implementacji. Udogodnienia te, eksportowane przez pseudomoduł SYSTEM, opisano w pierwszej części artykułu.

Dla procedury IOTRANSFER istotna jest możliwość blokowania pewnych przerwań na czas wykonywania określonych operacji. Dlatego każdy moduł ma określony priorytet, podobnie jak urządzenia zewnętrzne, które mogą powodować przerwania. Wykonywanie programu może być przerwane wtedy i tylko wtedy, gdy priorytet urządzenia jest wyższy od priorytetu modułu aktualnie wykonywanego. Priorytety urządzeń określone są sprzętowo, natomiast priorytet modułu (0..7) jest podawany w jego nagłówku.

Oprócz obiektów eksportowanych przez SYSTEM istnieją dwa dodatkowe udogodnienia, których charakter jest systemowo zależny. Pierwszym z nich jest możliwość użycia identyfikatora typu T jako funkcji konwersji typu argumentu na typ T. Działanie takiej funkcji zależy od reprezentacji danych w konkretnej implementacji. Drugie udogodnienie używane w deklaracjach zmiennych, umożliwia podawanie bezwzględnego adresu zmiennej. Adres ten zamknięty w nawiasy kwadratowe [] występuje po identyfikatorze deklarowanej zmiennej.

JEDNOSTKI KOMPILACJI

Jednostka kompilacji jest tekstem akceptowanym przez kompilator. Istnieją trzy rodzaje jednostek kompilacji: moduł główny, moduł definicyjny i moduł implementacyjny. Moduł główny zawiera program główny i tzw. moduły programowe. Moduł główny nie ma listy eksportowej. Obiekty importowane są definiowane w innych (oddzielnie kompilowanych) częściach programu, które z kolei składają się z dwóch części — modułu definicyjnego i implementacyjnego.

Moduł definicyjny określa nazwy i właściwości obiektów dostępnych dla klientów (ang. clients) tzn. modułów, które te obiekty importują. Moduł implementacyjny zawiera obiekty lokalne, których klient nie musi znać. Moduł definicyjny zawiera listę eksportową, deklaracje stałych, typów i zmiennych oraz specyfikacje nagłówków procedur. Odpowiadający mu moduł implementacyjny zawiera kompletne deklaracje procedur oraz inne deklaracje, które nie są eksportowane. Moduł definicyjny i implementacyjny istnieją wspólnie. Oba mogą zawierać listy importowe, a wszystkie obiekty zadeklarowane w module definicyjnym są dostępne w module implementacyjnym, bez jawnego ich importowania.

Definicja typu w module definicyjnym może zawierać pełną specyfikację typu. W takim przypadku eksport nazywany jest przezroczystym (ang. transparent). Jeśli w module definicyjnym występuje tylko identyfikator, a specyfikacja typu znajduje się w module implementacyjnym, eksport jest nieprzezroczysty (ang. opaque). Klient ma wtedy dostęp tylko do nazwy, a wszystkie właściwości typu są ukryte. Wszystkie procedury działające na argumentach tego typu muszą być również zdefiniowane w module implementacyjnym, który zawiera jego specyfikację. Eksport nieprzezroczysty jest dozwolony dla wskaźników i okrojeń typów standardowych.

FRAGMENT SKŁADNI MODULI-2

```
InstrukcjaIF = IF wyrażenie THEN CiągInstrukcji
  {ELSIF wyrażenie THEN CiągInstrukcji}
  [ELSE CiągInstrukcji] END.
InstrukcjaCASE = CASE wyrażenie OF wybór {"|" wybór}
  ELSE CiągInstrukcji END.
wybór = ListaEtykiet ":" CiągInstrukcji.
InstrukcjaWHILE = WHILE wyrażenie DO CiągInstrukcji END.
InstrukcjaREPEAT = REPEAT CiągInstrukcji UNTIL wyrażenie.
Instrukcja FOR = FOR identyfikator ":" wyrażenie TO wyrażenie
  [BY WyrażenieStałe] DO CiągInstrukcji END.
```

InstrukcjaLOOP = LOOP CiągInstrukcji END.
InstrukcjaWITH = WITH desygnator DO CiągInstrukcji END.
DeklaracjaProcedury = NagłówekProcedury ";" blok identyfikator.
NagłówekProcedury = PROCEDURE identyfikator [ParametryFormalne].
blok = {deklaracja} [BEGIN CiągInstrukcji] END.
DeklaracjaModułu =
MODULE identyfikator [priorytet] ";" {import} [eksport]blok
eksport = EXPORT [QUALIFIED] ListaIdentyfikatorów ";".
import = [FROM identyfikator] IMPORT Lista Identyfikatorów ";".
ModułDefinicyjny = DEFINITION MODULE identyfikator ";" {import}
[eksport] {definicja} END identyfikator ".".
ModułProgramu =
MODULE identyfikator [priorytet] ";" {import} blok identyfikator.
JednostkaKompilacji = ModułDefinicyjny :
[IMPLEMENTATION] ModułProgramu ".".

LITERATURA

[1] Iglewski M., Madey J., Matwin S.: PASCAL. Język wzorcowy — PASCAL 6000. WNT Warszawa, 1979
[2] Wirth N.: MODULA-2. Eidgenossische Technische Hochschule, Institut für Informatik, Zürich, March 1980.

OD REDAKCJI. Ze względu na znaczne ograniczenie miejsca, zmuszeni byliśmy istotnie skrócić bardzo obszerny materiał dostarczony przez Autora, wskutek czego powyższa treść artykułu nie wyczerpuje tematu. Dotyczy to szczególnie zapisu składni języka, za co Autora i Czytelników serdecznie przepraszamy. Pełny zapis składni — dostarczony przez Autora — prześlemy pocztą wszystkim zainteresowanym.

EWA SERAFIN
Wydział Transportu WSI
WIESŁAW SERAFIN
Zakłady Metalowe ŁUCZNIK
Radom

Programowy tester jednostek pamięci na dyskach elastycznych PLx45D.5

Jednostka pamięci na dyskach elastycznych typu PLx45D.5, produkowana przez Krakowską Fabrykę Aparatów Pomiarowych MERA-KFAP, stosowana jest głównie jako uniwersalna pamięć dla systemów mini- i mikrokomputerowych. Jej eksploatacja wymaga okresowej regulacji ustawienia głowic, wymiany zużytych głowic, wymiany zespołu centrującego dysku itp. Do przeprowadzania tych czynności producent zaleca tester walizkowy TSP-45A, ale ze względu na cenę (220 tys. zł, w sierpniu 1983) oraz trudności z zakupem — może być on zastąpiony testerem programowym, który pozwala wykonać wszystkie konieczne czynności serwisowe podczas eksploatacji. Naprawa za pomocą programowego testera nie wymaga demontażu jednostki PLx45D.5, co jest konieczne w przypadku użycia testera TSP-45A.

Charakterystyka pamięci PLx45D.5

Jednostka pamięci PLx45D.5 może współpracować z dwoma dyskami elastycznymi umieszczonymi w dwóch komorach (prawej i lewej). Dokonuje ona zapisu lub odczytu danych z obu stron dysku (A i B) w technice podwójnej częstotliwości, bez powrotu do zera. Głównymi elementami tej jednostki są: pakiet logiki oraz układ napędu. Okresowej regulacji wymaga układ napędu, który składa się z: silnika synchronicznego, silnika krokowego — nadającego ruch posuwny pozycjonerowi za pośrednictwem śruby, oraz pozycjonera z przymocowanymi do niego dwiema głowicami zapisu i odczytu danych z dysku.

Do najczęstszych uszkodzeń układu napędowego należą:
— nieprawidłowe ustawienie głowic względem pozycjonera
— mała rozdzielczość zużytych głowic
— zbyt duży magnetyzm szczątkowy głowicy
— zużycie zespołu centrującego, powodujące „bicie" dysku
— zużycie zespołu: śruba silnika krokowego — pozycjoner.

Pakiet logiki PLx45D.5 zawiera:
— wzmacniacz zapisu
— wzmacniacz odczytu
— układ zegarowy
— układy sterujące pozycjonerem głowic
— układy sterujące napędem dysków
— obwody kontrolujące prawidłowe warunki pracy dysku.

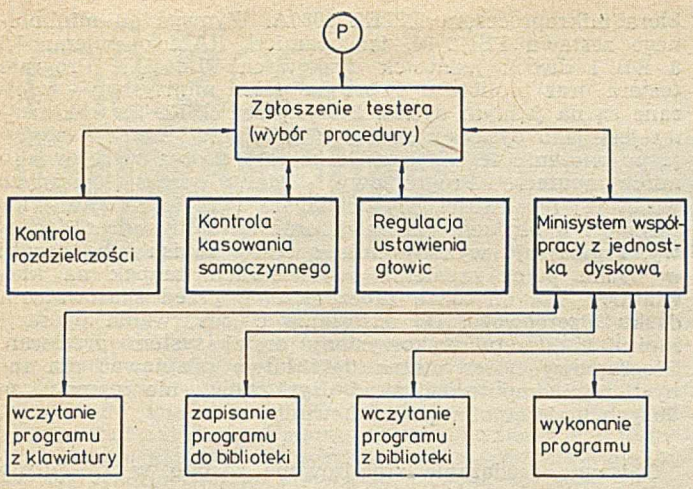
Pakiet ten wyposażony jest tylko w jeden element regulacyjny — potencjometr ustalający częstotliwość układu zegarowego. Regulacja wykonywana jest w zasadzie tylko po wymianie niektórych elementów w pakiecie.

Charakterystyka programowego testera

Programowy tester jednostek pamięci PLx45D.5 jest zbiorem procedur realizujących funkcje zapisu i odczytu z dysku w taki sposób, aby umożliwić wykonanie czynności serwisowych. Jego strukturę przedstawia rysunek. Po zgłoszeniu się testera (wyświetlenie identyfikatora i zestawu funkcji na ekranie monitora) możliwy jest wybór jednej z czterech przedstawionych na rysunku procedur. Procedury te są niezależne i mogą być wykonywane w dowolnej kolejności, chociaż zalecany jest porządek podany na rysunku (od lewej).

Mgr inż. EWA SERAFIN ukończyła studia w 1979 r. na Wydziale Elektroniki Politechniki Warszawskiej, w Instytucie Informatyki. Obecnie pracuje na stanowisku asystenta w Instytucie Eksploatacji Taboru Kolejowego, na Wydziale Transportu Wyższej Szkoły Inżynierskiej w Radomiu. Zajmuje się komputerowymi układami diagnostyki oraz zastosowaniem języka naturalnego w systemach komputerowych.

Mgr inż. WIESŁAW SERAFIN ukończył studia w 1980 r. na Wydziale Elektroniki Politechniki Warszawskiej, w Instytucie Automatyki. Obecnie pracuje w Ośrodku Przetwarzania Informacji Zakładów Metalowych ŁUCZNIK im. Gen. Waltera w Radomiu na stanowisku specjalisty elektronika. Zajmuje się budową i oprogramowaniem minikomputerów oraz cyfrowymi układami sterowania.



Struktura programowego testera

Kontrola rozdzielczości

Rozdzielczość głowicy obliczana jest ze wzoru:

$$R = \frac{A(1)}{A(0)} \cdot 100\ \%$$

gdzie:

R — rozdzielczość wyrażona w procentach
A(1) — Średnia amplituda sygnału ze wzmacniacza odczytu przy odczycie bajtów równych FF_{16} (wszystkie bity informacyjne na ścieżce są równe **1**)

A(0) — średnia amplituda sygnału ze wzmacniacza odczytu przy odczycie bajtów równych 00_{16} (wszystkie bity informacyjne na ścieżce są równe **0**).

Głowicę można uznać za dobrą, jeżeli rozdzielczość dla wszystkich ścieżek jest większa niż 35%. W praktyce wystarczy dokonać pomiaru dla ścieżki 76, dla której rozdzielczość jest najmniejsza (ze względu na największą gęstość zapisu). Do pomiaru rozdzielczości przewidziane zostały następujące funkcje:

1. Zapisanie ścieżki 76 bajtami równymi 00_{16}
2. Zapisanie ścieżki 76 bajtami równymi FF_{16}
3. Odczyt informacji ze ścieżki 76.

Użytkownik wybiera jedną z trzech powyższych możliwości (kolejność dowolna) podając jej numer. W czasie realizacji funkcji 3 (odczyt) należy zmierzyć za pomocą oscyloskopu amplitudę sygnału ze wzmacniacza odczytu (punkty TP1 i TP2 na pakiecie logiki), a następnie obliczyć rozdzielczość **R**.

Kontrola kasowania samoczynnego

Kasowanie samoczynne spowodowane jest magnetyzmem szczątkowym głowicy. Powoduje to niszczenie zapisanej informacji na dysku, mimo braku prądu w uzwojeniu zapisu. Poziom magnetyzmu szczątkowego obliczany jest ze wzoru:

$$K = \left(1 - \frac{A1}{A2}\right) \cdot 100\ \%$$

gdzie:

K — współczynnik charakteryzujący magnetyzm szczątkowy głowicy
A1 — pierwotna amplituda sygnału ze wzmacniacza odczytu
A2 — amplituda sygnału ze wzmacniacza odczytu po przesunięciu głowicy.

Producent dopuszcza maksymalną wartość **K = 15%**. Jeżeli **K** jest większa niż 15%, głowicę należy wymienić. W celu pomiaru **K** tester wykonuje następujące operacje:
- zapisanie ścieżki 38 bajtami równymi FF_{16}
- odczyt ścieżki 38 — pomiar **A1**
- przesunięcie głowicy w okolicach ścieżki 38 (w przód i w tył) — czynność wykonywana ręcznie
- odczyt ścieżki 38 — pomiar **A2**.

Poziom kasowania samoczynnego jest jednakowy na wszystkich ścieżkach, a ścieżka 38 została wybrana zgodnie z zaleceniem producenta.

Regulacja ustawienia głowic

Do regulacji ustawienia głowic niezbędny jest wzorcowy dysk położenia (BASF CE-Flexy Disk). W czasie odczytu informacji ze ścieżki 36 dysku wzorcowego należy obserwować sygnał ze wzmacniacza odczytu. Głowicę należy przesuwać względem pozycjonera — aż do uzyskania przebiegu zgodnego z podanym w [1]. Ewentualne trudności z uzyskaniem podanego przebiegu mogą być związane z uszkodzeniem zespołu centrującego.

Po zakończeniu regulacji położenia głowic, należy ocenić jakość układu: śruba silnika krokowego — pozycjoner. Odbywa się to przez obserwację sygnału odczytanego ze ścieżki 36 przy osiąganiu jej przez głowicę od strony ścieżek o niższych numerach, a następnie o wyższych. Przebieg ze wzmacniacza odczytu w obu przypadkach nie powinien się zmieniać. Zmiany świadczą o zużyciu układu: śruba silnika krokowego — pozycjoner. Niewielkie luzy można kasować zmieniając naprężenie sprężyny w zespole pozycjonera.

Tester wykonuje kolejno następujące operacje:
— odczyt ze ścieżki 36
— przesunięcie pozycjonera na ścieżkę 35, a potem odczyt z 36
— przesunięcie pozycjonera na ścieżkę 37, a potem odczyt z 36
— przesunięcie pozycjonera na ścieżkę 26, a potem odczyt z 36
— przesunięcie pozycjonera na ścieżkę 46, a potem odczyt z 36.

Ze względu na specyficzne informacje zapisane na dysku wzorcowym podprogram odczytu nie może zawierać porównania numeru ścieżki odczytanego z nagłówka ścieżki z liczbą kroków silnika (sygnalizowany jest błąd nie odnalezienia ścieżki). Osiąganie poszczególnych ścieżek odbywa się przez zliczanie kroków silnika krokowego. Podane podprogramy wykonywane są kolejno.

Dodatkowo, tester ma możliwość odczytu ze ścieżki 0 i 76 dysku wzorcowego. Wykorzystywane jest to do wstępnego ustawiania głowic po demontażu zespołu pozycjonera (np. po wymianie głowicy). Zamiast dysku wzorcowego może być wykorzystany inny dysk, z zapisanymi ścieżkami 0, 36 i 76 na dobrze wyregulowanej jednostce dyskowej. Głowice ustawiamy wtedy w położeniu dającym maksymalną amplitudę sygnału ze wzmacniacza odczytu. Pozwala to na przybliżoną regulację bez użycia dysku wzorcowego.

Minisystem współpracy z jednostką dyskową

Opisane procedury pozwalają na wykonanie typowych czynności serwisowych. W celu rozszerzenia możliwości testera, oprogramowano pewne funkcje pamięci Plx45D.5 Podstawą było opracowanie następującej listy instrukcji (w nawiasach — długość instrukcji w bajtach):

0 — wczytanie numeru ścieżki i sektora (3)
1 — odczytanie informacji z sektora do bufora (1)
2 — zapisanie sektora informacją z bufora (1)
3 — odczytanie całej ścieżki, w buforze pamiętany jest ostatni odczytany sektor (1)
4 — zapisanie 26 kolejnych sektorów identyczną informacją z bufora (1)
5 — inkrementacja numeru sektora (1)
6 — inkrementacja numeru ścieżki (1)
7 — dekrementacja numeru sektora (1)
8 — dekrementacja numeru ścieżki (1)
9 — wypełnienie bufora jednym znakiem, szesnastkowy kod znaku wczytywany jest z klawiatury (1)
10 — wypełnianie bufora ciągiem 128 znaków wczytywanych z klawiatury (1)
11 — skok bezwarunkowy (2)
12 — skok warunkowy w zależności od naciśnięcia określonego klawisza (2)
13 — otwarcie drzwi komór dyskowych (1)
14 — stop (1).

Maksymalna długość programu wynosi 128 bajtów. Numer instrukcji jest jednocześnie jej kodem. Rozkaz **0** wczytuje dwie następne liczby i traktuje je jako numer

ścieżki i numer sektora. Instrukcje 11 i 12 interpretują następną liczbę jako numer instrukcji programu, do której wykonywany jest skok. Instrukcje wczytywane są z klawiatury i muszą być oddzielone jakimkolwiek znakiem nienumerycznym (np. spacją). W celu realizacji napisanych programów został opracowany minisystem współpracy z jednostką dyskową o następujących funkcjach:

1. Wczytanie programu z klawiatury.
2. Zapisanie programu do biblioteki.
3. Wczytanie programu z biblioteki.
4. Wykonanie programu.

Funkcja 1 realizuje wczytanie programu do pamięci operacyjnej. Program ten może być zapisany do biblioteki zrealizowanej na dysku systemowym zawierającym tester (funkcja 2). Funkcja 3, oprócz odczytu programu z biblioteki, pozwala nanieść do niego poprawki. Identyfikatorem programu w bibliotece jest jego adres dyskowy (numer ścieżki i sektora), co znacznie uprościło tester. Wykonanie programu znajdującego się w pamięci operacyjnej (wczytany funkcją 1 lub 3) polega na kolejnym wywoływaniu podprogramów realizujących poszczególne instrukcje.

Przedstawiony minisystem został zaprojektowany tak, aby umożliwić pisanie programów współpracy z dyskami nawet dla osób o małym doświadczeniu. Starano się maksymalnie uprościć kody i zakres listy instrukcji, aby tej części testera nadmiernie nie rozbudowywać. Podany zestaw instrukcji jest wystarczający do celów naprawy jednostek pamięci PLx45D.5. Pozwala dokonywać napraw układu napędowego, a także pakietu logiki. Zachowana została możliwość łatwego rozszerzenia zbioru instrukcji.

Uwagi eksploatacyjne

Programowy tester jednostek pamięci PLx45D.5 zrealizowany został dla minikomputera PSPD-90 w języku asemblera mikroprocesora INTEL 8080A. Wymaga on minimalnego zestawu PSPD-90, tzn. pamięci RAM o pojemności 8 KB i dwóch jednostek dyskowych PLx45D.5. Program testera oraz biblioteka tworzona przez minisystem— zapisane są na jednym dysku, który przez PSPD-90 traktowany jest jako dysk systemowy. Wywołanie testera następuje po włożeniu tego dysku do komory 0 oraz wyzerowaniu minikomputera. Programowy tester, oprócz opisanych wcześniej możliwości, wyświetla na ekranie monitora informacje o błędach transmisji oraz aktualny adres dyskowy, sygnalizuje wykonywanie operacji zapisu lub odczytu w dysku przez zapalenie odpowiednich lampek na klawiaturze. Ma on także zabezpieczenie przed skasowaniem dysku wzorcowego. Do określania czasu trwania operacji zapisu i odczytu wykorzystany został system przerwań. Programowy tester można dość łatwo adaptować dla innych minikomputerów, a od użytkownika nie wymaga on dużych umiejętności programowania.

Opisane rozwiązanie zrealizowane zostało w Zakładach Metalowych ŁUCZNIK w Radomiu. Tester całkowicie potwierdził swoją przydatność do napraw i regulacji jednostek pamięci PLx45D.5 w czasie kilkumiesięcznej intensywnej ekpsloatacji (od czerwca 1983).

LITERATURA

[1] BASF CE-Flexy Disk 200 mm for track adjustment, amplitude control and index sensor alignment. Spezifikation und Description, BASF Aktiengesellschaft, November 1980

[2] Jednostka pamięci na dyskach elastycznych typ PLx45D.5 — DTR, MERA—KFAP, Kraków.

Przegląd języków wysokiego poziomu (4)

Języki nowoczesne

Czwarta część „Przeglądu" zawiera omówienie języków, które często określa się jako nowoczesne. Określenie to jest stosowane prawdopodobnie nie tylko ze względu na okres, w którym zaczęto je rozwijać, ale także z innych powodów, np. ze względu na próby rozwiązania problemu weryfikacji lub z uwagi na nacisk, jaki kładą na ogólność danych. Zostaną podane jedynie krótkie wprowadzenia w problematykę każdego z języków tej kategorii (z wyjątkiem języka ADA, któremu poświęcono więcej miejsca), ze względu na przekonanie autorki, że będzie on miał duże znaczenie dla programowania [1]). Charakterystyki języków przedstawiono chronologicznie, zgodnie z czasem ich powstania.

ALGOL 68, PASCAL, CLU, EUCLID oraz SETI

Oprócz wymienionych powyżej istnieją jeszcze dwa języki, które zasługują choćby na krótką wzmiankę, a mianowicie SIMULA 67 (Dahl i Nygaard, 1967) oraz ALPHARD (Shaw i in., 1977).

SIMULA 67 została pominięta jako język już zbyt „stary". Jednak warto sobie uświadomić, że wprowadzone w tym języku pojęcie „klasy" miało duży wpływ na współczesne prace nad ogólnością danych. Zarówno twórcy języka ALPHARD, jak i CLU, przyznają, że wiele pomysłów zawdzięczają koncepcjom SIMULI 67.

Mimo że na temat języka ALPHARD napisano wiele, to jednak nie ma on żadnej implementacji i z tego względu jego opis pominięto.

[1]) Na łamach INFORMATYKI były już przedstawione podstawowe zasady programowania języka ADA. (Zalewski J.: ADA — nowy język programowania, nr 11—12/81, 1, 2, 3, 4—5, 6/82) — przyp. red.

Używane są jeszcze dwa nowoczesne języki, które nie zostaną omówione, a mianowicie MESA opisana przez Geschke i in. (1977) oraz MODULA opisana przez Wirth'a (1977).

ALGOL 68

Nie upłynęło zbyt wiele czasu od wprowadzenia ALGOLU 60, kiedy już zaczęto interesować się jego rozszerzaniem. Około 1968 roku Van Wijngaarden oraz kilka innych osób opracowało w Europie język ALGOL 68. Język ten w ostatecznym swym kształcie nie jest jednak rozszerzeniem ALGOLU 60, lecz — według autorki — ostatnim ze „starych, potężnych' języków. Porównywalny rozmiarami i możliwościami jest opracowany około 1964 roku PL/I, który zawierał wiele nowych rozwiązań. Ponieważ jednak ALGOL 68 jest mniej znany, a wprowadził pewne bardzo interesujące pomysły, warto go dokładniej scharakteryzować.

Jedną z kluczowych koncepcji ALGOLU 68 jest ortogonalność. Starano się w nim zmniejszyć liczbę niezależnych pojęć pierwotnych, wprowadzając zaledwie niewielką liczbę podstawowych konstrukcji i reguł systematycznego i regularnego ich składania. ALGOL 68 jest językiem o strukturze blokowej i jest rozszerzalny. Ma bogaty zbiór typów danych, a ponadto programista dysponuje środkami do definiowania nowych typów danych. Jest językiem wyrażeń, co oznacza, że każdej konstrukcji języka przypisuje się wartość — i z tego względu może ona pojawiać się z prawej strony instrukcji podstawienia. Dotyczy to także tablic i struktur, a dzięki indeksatorowi, który jest uogólnieniem pojęcia wskaźnika (z ALGOLU 60), istnieje możliwość otrzymywania podtablic oryginalnych tablic.

Ciąg dalszy na str. 21

Nie mamy przed sobą jeszcze wizji zinformatyzowanej Polski, kraju, w którym każda pojedyncza inicjatywa, każda praca mogłaby być wspierana (mikro)komputerem, narzędziem efektywniejszym od poprzednich. To przyszłość trudna do wyobrażenia. Niemniej dostrzegamy już sporo przejawów zmiany. Zmiany sposobu myślenia o informatyce, o własnym w niej udziale. Informatyka przestaje być u nas rzeczywistością samoistną, zaczyna wreszcie zbliżać się do życia — indywidualnego, zbiorowego... Tak, konstatacja ta nie jest powszechna, a nasz KLAN ma w istocie wymiar mikro. Ale zaczyna być klanem ludzi rozumianych...

Pamięci kasetowe stają się przeżytkiem. W krajach o przeciętnie rozwiniętej technice komputerowej stosowane są już tylko w prymitywnym sprzęcie, spełniającym rolę „inteligentnej" zabawki.

Całkiem odmiennie wygląda ta sprawa u nas. Magnetofon kasetowy jest w zasadzie jedynym ratunkiem dla hobbystów, ba — często nawet dla profesjonalnych użytkowników mikrokomputerów. O pamięciach domenowych, EAROM-ach (ang. Electricaly Alterable Read Only Memory) trudno nawet marzyć. Nabycie stacji dysków elastycznych (nawet tej Made in Poland) przedstawia się nader mgliście. W ciągu ostatniego roku zetknąłem się tylko z jednym szczęśliwcem, któremu udało się tego dokonać. Ale skoro mógł on zainstalować komputer ODRA na ciężarówce i sterować czymś tam na poligonie, to i w potyczkach „o dyski" miał zapewne specjalny priorytet.

W mikrokomputerach dla zapisu informacji na taśmie magnetycznej wykorzystywane są zazwyczaj metody o „filozofii" zbliżonej do kodu Kansas City. W realizacji sprzętowej toru odczytu wykorzystuje się dwa filtry częstotliwościowe. Na wyjściu jednego pojawia się sygnał dla „1", na wyjściu drugiego dla „0".

Poniżej zamieszczamy opis rozwiązania, w którym odczyt i zapis informacji zrealizowano całkowicie programowo. Koszt konstrukcji podraża niestety konieczność zastosowania programowanego licznika (nie produkowany w kraju). Opisane rozwiązanie wykorzystane zostało w systemie MSA80.1000 opracowanym w Przemysłowym Instytucie Elektroniki w Warszawie i wdrażanym obecnie do produkcji w zakładach ZUMPE w Szczytnie. Ponieważ w tym systemie mikroprocesor równocześnie obsługuje klawiaturę i wyświetlacze cyfrowe, prędkość transmisji wynosi tylko 150 bodów. Jeżeli równolegle z obsługą magnetofonu mikroprocesor nie musi realizować innych zadań, można tę prędkość zwiększyć.

Programowa obsługa pamięci kasetowej

Podstawowym problemem przy przechowywaniu informacji cyfrowej na taśmie magnetycznej jest wybranie takiego sposobu kodowania, który pozwala osiągnąć możliwie dużą odporność na zakłócenia i przekłamania. Z reguły konieczny jest pewien kompromis między niezawodnością a szybkością przesyłanej informacji. W proponowanym rozwiązaniu stanowi logicznemu „0" odpowiada osiem impulsów o częstotliwości 1200 Hz, a stanowi logicznemu „1" — cztery impulsy o częstotliwości 600 Hz. Szybkość transmisji wynosi w tym przypadku 150 bodów. Transmisja jest asynchroniczna, z dwoma bitami stopu.

Organizacja logiczna przesyłanych zbiorów może być zrealizowana różnymi sposobami i nie będzie tu analizowana. Opisana zostanie natomiast realizacja zapisu i odczytu informacji na poziomie bitu.

W przedstawianym rozwiązaniu do współpracy z mikrokomputerem można wykorzystać magnetofon kasetowy MK

ZWIASTUN

Rozmowa z LESZKIEM WILKIEM, prezesem Klubu Mikrokomputerowego ABAKUS

μK: — Sytuacja kreuje Pana na przywódcę polskiej mikroinformatyki. Niemal zewsząd słyszymy Pana nazwisko...

Leszek Wilk: — Może dlatego, że jestem jednym z niewielu, którzy myślą o mikroinformatyce w kategoriach społecznych. Jakieś piętnaście lat temu zrozumiałem, że komputer zapowiada rewolucję. Przygotowywałem się. Czekałem, aż będzie w moim zasięgu. Dopiero w ciągu ostatnich dwóch lat ceny mikrokomputerów tak spadły, że mogłem za własne pieniądze kupić ZX81 — i wystartować.

Zainteresowanie mikroinformatyką w Polsce jest ogromne. Świadczą o tym rozmowy z ludźmi. I fakt, że coraz więcej osób posiada własny sprzęt. Ja sam zainteresowany jestem najbardziej wykorzystaniem komputerów do edukacji. Wprawdzie nie jestem informatykiem, ale miałem sprzęt i pomysły — poszukałem więc innych. Tak powstał prywatny klub, początkowo u mnie w domu.

Leszek Wilk (z lewej) w pojedynku z komputerem

μK: — A skąd pierwsi klubowicze?

L.W.: — Koledzy lub ludzie z ogłoszenia, które dałem do gazety. Ale zgłosili się głównie profesjonaliści. A z profesjonalistami pracuje się dość ciężko. Oni są w stanie zrobić poważne, rozbudowane programy — translatory, kompilatory, etc. Natomiast zastosowań, które bezpośrednio służyłyby człowiekowi, nie specjaliście, jakoś nie potrafią realizować. Nie mają tego w sobie.

Ale gdybym nie ja zaczął — byłby ktoś inny. To się musi lada chwila zacząć na dobre. Musi eksplodować. Pierwszy krok zrobią ogólnodostępne kluby. W dzisiejszej sytuacji — przy braku sprzętu i oprogramowania, ale przy sporym zainteresowaniu — to dla nas jedyna droga. A zaczynać trzeba szybko.

μK: — To jest poczucie obowiązku czy misji?

L.W.: — Jestem ekonomistą. Napisałem kiedyś pracę we Francji, po francusku. Omawia obecną sytuację ekonomiczną świata i na tym tle — ekonomię Polski, a w szczególności handlu zagranicznego. Wnioskiem z tej pracy może być stwierdzenie, że motorem rewolucji, jaka

Zwiastun

odbywa się obecnie w ekonomii światowej, jest mikroprocesor i że inwestycje powinny pójść już dawno w zupełnie innym kierunku, co oznacza — między innymi — ogromną szansę dla Polski. To zresztą jest jednocześnie szansa i przymus. Ekonomia światowa dematerializuje się, tzn. zaczyna opierać się na dwóch głównych filarach: informacji i edukacji. Mówi zresztą o tym ostatni Raport Rzymski. Z drugiej strony — do obecnego poziomu cywilizacyjnego człowiek po prostu nie dorasta. Nie dlatego, że nie potrafi — ma w końcu ogromne możliwości, w małym tylko stopniu wykorzystywane — ale dlatego, że nie jest odpowiednio wyedukowany. Trzeba więc szukać metod, abyśmy mogli dorosnąć do warunków, jakie zostały już stworzone, abyśmy mogli przeskoczyć do tej nowej, oczekującej nas — cywilizacji informacyjno-edukacyjnej. Byłem uczniem, studentem, a potem nauczycielem w szkole — i jestem przekonany, że jest to możliwe tylko z pomocą mikrokomputerów.

µK: — Ale Pan nie rozróżnia kryzysu ekonomii światowej od kryzysu u nas, jednak wyraźnie odmiennego.

L.W.: — Nieprawda. W gruncie rzeczy problemy są te same. Byłem we Francji, i znam jej gospodarkę. Mimo różnicy ustrojów, jest wiele identycznych problemów. Za dwa—trzy lata będą mieć te same trudności, które my mieliśmy w roku 1979.

µK: — Tam jednak trwa boom informatyczny, a u nas...

L.W.: — Oni też niewiele zrobili. Owszem, jest tam ileś set tysięcy mikrokomputerów. Ale jaki jest z nich pożytek?! To są w przewadze ogłupiające gry. W Europie dotychczasowe oprogramowanie „edukacyjne" to właściwie nic wobec ogromu potrzeb — jakieś pojedyncze, przypadkowe programy, zamiast wielotysięcznego rynku. Lepiej jest tylko w Stanach Zjednoczonych.

Ekonomia „materialna", rozwinięta na Zachodzie, która za swój główny cel ma wzrost produkcji, już się kończy. Tak jak kiedyś gwałtownie zmniejszyło się znaczenie rolnictwa na rzecz przemysłu, tak teraz przemysł ustąpi miejsca edukacji. Ale Zachód bardzo wolno zmienia kurs. Mamy więc szansę ich dogonić. Zamiast bawić się teraz w produkcję rzeczy materialnych, których zresztą nie umiemy produkować (z jednej tony surowca wytwarzamy znacznie mniej niż na Zachodzie), możemy od razu przeskoczyć do „niematerialnej". Mamy w niej większe szanse.

µK: — Zadziwiający jest ten Pana optymizm. Jak my, przy naszych zaściankowych umysłach, kiepskiej edukacji, z poczuciem bezradności w wielu sferach życia zbiorowego, mamy tego dokonać?

L.W.: — Najpierw trzeba przygotować ludzi, A ludzie nie są przystosowani do wymogów dnia dzisiejszego — tu, i tam. Przeciętnie to jest ten sam poziom. Tak więc barierę stanowi konieczność zdecydowanej inwestycji w edukację. My możemy to zrobić wcześniej. Umysły „zachodnie", opanowane zasadami starej ekonomii, nie są w stanie dokonać szybkiego przewrotu. Oni myślą kategoriami produkcji krótko- i średniookresowej, kryteriami zysku. My, nie mając nic do stracenia, możemy myśleć perspektywicznie. Teoretycznie biorąc, nam łatwiej byłoby zainwestować w masową edukację.

µK: — No dobrze, niebawem będzie miał Pan w Klubie tysiąc osób przy mikrokomputerze. I co dalej?

L.W.: — Nie! To nie jest tak. Przyszłość mikroinformatyki nie jest już dzisiaj zależna od sprzętu, lecz od oprogramowania. Wartość sprzętu stanowi jakieś 10—30% wartości całego komputera. A my możemy wyprodukować oprogramowanie...

µK: — Ale jak rozwijać oprogramowanie bez sprzętu? I to jeszcze masowo.

L.W.: — To prawda, nie można — w tej tzw. dużej informatyce, gdzie wymagany stopień komplikacji jest znacznie 232 (lub podobny). Magnetofon jest dołączony do mikrokomputera za pośrednictwem trzech linii sygnałowych — zapisu, odczytu i włączenia napędu. W celu zmniejszenia kosztów konstrukcji, wszystkie trzy linie obsługiwane są programowo. Układowo zrealizowano jedynie przetworzenie sygnałów o poziomach TTL na sygnały wchodzące lub wychodzące z magnetofonu. Dodatkowym elementem systemu mikroprocesorowego jest programowany licznik (np. 8253), generujący przerwanie po czasie zadanym przez procesor.

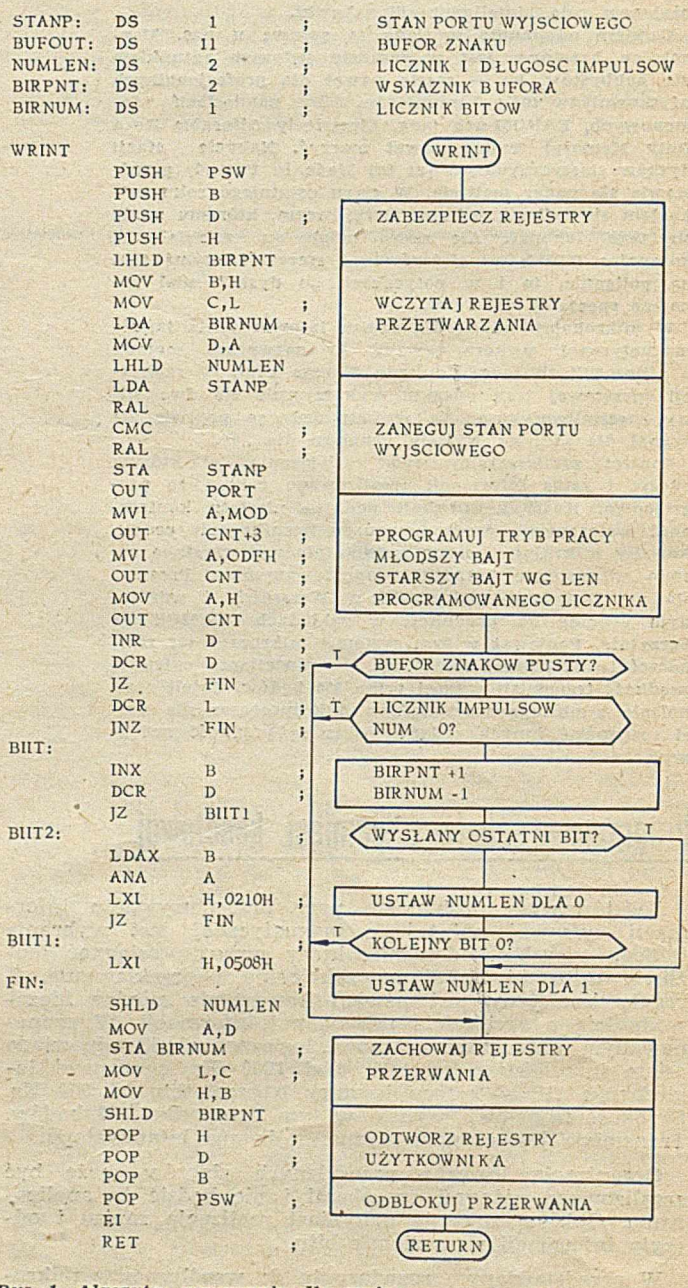
Zapis

Przy zapisie sygnału licznik generuje przerwanie co 835 µs dla bitu „1" lub co 417 µs w przypadku bitu „0". Program obsługi przerwania wykorzystuje 15 bajtów pamięci RAM, w których przechowuje swoje zmienne.

BUFOUT o długości 10 bajtów przeznaczony jest na przechowywanie aktualnie wyprowadzanego znaku. Każdemu bitowi znaku przyporządkowano jeden bajt w bufo-

```
STANP:  DS    1          ;   STAN PORTU WYJSCIOWEGO
BUFOUT: DS    11         ;   BUFOR ZNAKU
NUMLEN: DS    2          ;   LICZNIK I DLUGOSC IMPULSOW
BIRPNT: DS    2          ;   WSKAZNIK BUFORA
BIRNUM: DS    1          ;   LICZNIK BITOW

WRINT                    ;
        PUSH  PSW
        PUSH  B
        PUSH  D          ;   ZABEZPIECZ REJESTRY
        PUSH  H
        LHLD  BIRPNT
        MOV   B,H
        MOV   C,L        ;   WCZYTAJ REJESTRY
        LDA   BIRNUM     ;   PRZETWARZANIA
        MOV   D,A
        LHLD  NUMLEN
        LDA   STANP
        RAL
        CMC              ;   ZANEGUJ STAN PORTU
        RAL              ;   WYJSCIOWEGO
        STA   STANP
        OUT   PORT
        MVI   A,MOD
        OUT   CNT+3      ;   PROGRAMUJ TRYB PRACY
        MVI   A,ODFH     ;   MŁODSZY BAJT
        OUT   CNT        ;   STARSZY BAJT WG LEN
        MOV   A,H        ;   PROGRAMOWANEGO LICZNIKA
        OUT   CNT        ;
        INR   D          ;
        DCR   D          ;   T BUFOR ZNAKOW PUSTY?
        JZ    FIN
        DCR   L          ;   T LICZNIK IMPULSOW
        JNZ   FIN        ;     NUM  0?
BIIT:
        INX   B          ;   BIRPNT +1
        DCR   D          ;   BIRNUM -1
        JZ    BIIT1
BIIT2:                   ;   WYSŁANY OSTATNI BIT?  T
        LDAX  B
        ANA   A
        LXI   H,0210H    ;   USTAW NUMLEN DLA 0
        JZ    FIN
BIIT1:                   ;   T KOLEJNY BIT 0?
        LXI   H,0508H
FIN:                     ;   USTAW NUMLEN DLA 1
        SHLD  NUMLEN
        MOV   A,D
        STA BIRNUM       ;   ZACHOWAJ REJESTRY
        MOV   L,C        ;   PRZERWANIA
        MOV   H,B
        SHLD  BIRPNT
        POP   H          ;   ODTWORZ REJESTRY
        POP   D          ;   UŻYTKOWNIKA
        POP   B
        POP   PSW        ;   ODBLOKUJ PRZERWANIA
        EI
        RET              ;   RETURN
```



Rys. 1. Algorytm przerwania dla zapisu

rze, dzięki czemu znacznie skrócono czas obsługi przerwania. Pierwszy bajt bufora znaku równy jest 00H (bit startu), a dwa ostatnie mają wartość 0FFH (dwa bity stopu). W zmiennej BIRPNT przechowywany jest wskaźnik aktualnie wyprowadzonego bitu. Zmienna BIRNUM zawiera licznik bitów, które pozostały do wyprowadzenia. Jeżeli osiągnie on wartość 0, automatycznie wyprowadzane będą impulsy o częstotliwości 600 Hz. Zmienna NUM zawiera informację o tym, ile przerwań zostanie przyjętych dla wyprowadzenia całego bitu (8 dla bitu „1" i 16 dla bitu „0"). Zamienna LEN przechowuje wartość, przepisywaną do programowanego licznika, określającą czas trwania kolejnego impulsu.

Algorytm obsługi przerwania dla zapisu przedstawia rysunek 1.

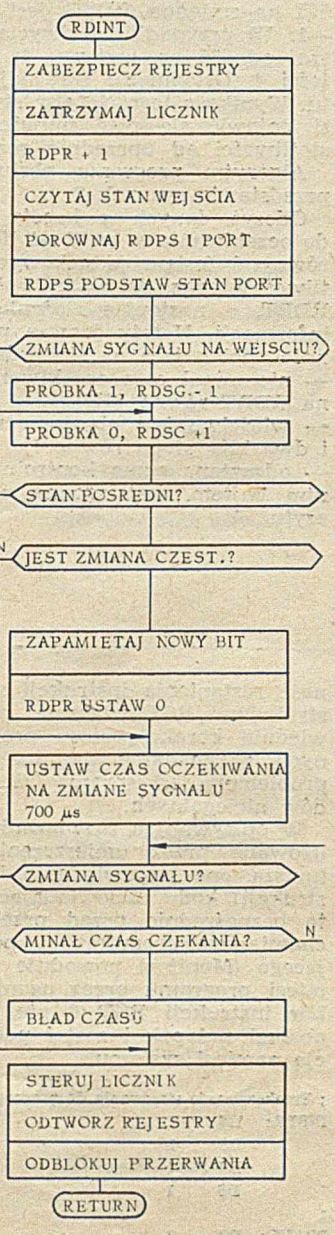
Dla użytkownika systemu zapis danych na taśmę magnetyczną sprowadza się do kolejnego wpisywania bajtów do bufora BUFOUT i oczekiwania w pętli, aż bajt zostanie

```
RDBT:   DS    1      ;   WARTOSC BITU
RDPR:   DS    1      ;   LICZNIK PROBEK
RDPS:   DS    1      ;   POCZATKOWY SYGNAL
RDSC:   DS    1      ;   LICZNIK SYGNALU

RDINT:                ;
        PUSH  PSW
        PUSH  H       ;
        MVI   A,MOD1
        OUT   CNT+3   ;
        LXI   H,RDPR  ;
        INR   M       ;
        IN    PORT
        CMA           ;
        ANI   80H
        INX   H       ;
        CMP   M
        MOV   M,A     ;

        INX   H       ;
        JZ    RDITO   ;
        DCR   M
        DCR   M       ;
RDITO:
        INR   M       ;
        MVI   A,0FFH
        JM    RDIT1
        MVI   A,04H   ;
        CMP   M       ;
        JNC   RDIT2
RDIT1:                ;
        MOV   M,A
        ANI   80H
        LXI   H,RDBT
        CMP   M       ;
        JZ    RDIT2
        MOV   M,A
        INX   H       ;
        MVI   M,00H
RDIT2:
        LDA   RDPS    ;
        MOV   L,A     ;
        MVI   H,30H   ;
RDIT3:
        IN    PORT
        ANI   80H     ;
        CMP   L
        JZ    RDIT4
        DCR   H       ;
        JNZ   RDIT3

        LXI   H,RDBT
        MVI   M,01H   ;
RDIT4:
        MVI   A,0
        OUT   CNT     ;
        MVI   A,4
        OUT   CNT     ;
        POP   H
        POP   PSW     ;
        EI
        RET           ;
```



Rys. 2. Algorytm programu obsługi przerwania dla odczytu

Zwiastun

większy. Ale nie w mikroinformatyce. Tu najważniejszy jest pomysł. Proszę przypomnieć sobie, jak rozwijała się mikroinformatyka na Zachodzie. To nie IBM, nie CDC ją rozwinęły, ale... dwuosobowa (na początku) APPLE. A w Europie? To przede wszystkim pojedynczy ludzie w Anglii, nie potentaci wielkiego przemysłu.

Jeśli natomiast w Polsce do końca tego roku powstanie dwadzieścia ogólnodostępnych klubów, takich jak mój, w największych miastach — to dla wszystkich twórczych ludzi starczy sprzętu, literatury i podstawowego oprogramowania.

µK: — Powiedzmy, że za rok będzie nawet więcej niż dwadzieścia otwartych klubów. I co dalej? W co może się — w wymiarze społecznym — przetworzyć ta praca kilkudziesięciu grup zapaleńców?

L.W.: — Powstaną dobre programy, głównie gry edukacyjne i programy do przetwarzania informacji (np. VisiCalc). W ciągu dziesięciu lat powinno ich powstać kilka, kilkanaście tysięcy.

µK: — I wykorzystywać je będą członkowie klubów...

L.W.: — Nie, one zostaną oddane Ministerstwu Oświaty, Ministerstwu Szkolnictwa Wyższego, przedszkolom, szkołom — wszystkim zainteresowanym. Będą sprzedawane wraz ze sprzętem, którego już wówczas będzie dużo. Jeśli mikrokomputer będzie kosztował — przy dzisiejszych cenach — 50—70 tys., to już będzie można mówić o dostępności. Nabytek taki może się nawet finansowo opłacać, gdy — powiedzmy — dzięki komputerowi będzie można nauczyć się jako tako języka obcego w kilkanaście tygodni.

µK: — Jaki ma być ten człowiek, którego chciałby Pan wyedukować?

L.W.: — Twórczy, myślący szerzej, szybko i efektywnie, nie jak dzisiaj. Dam przykład. Poszedłem do dyrekcji Domów Centrum z propozycją zorganizowania stoiska informacyjno-reklamowego związanego z usługami. Zostałem przyjęty na stojąco. Gdy powiedziałem o komputerach, ta pani mnie niemal wyrzuciła za drzwi. Gdybym był szewcem czy krawcem, to proszę — możemy rozmawiać. Oni chcą robić usługi i na nic innego nie mają czasu. Gdy właśnie głównym problemem usług jest niedostateczna informacja — kto, co i gdzie wykonuje, na jakich zasadach. Komputerowy system zbierający informacje o usługach Domów Centrum i wokół nich byłby niezwykle cenny.

W Krakowie zwróciłem się do Urzędu Miasta z propozycją uruchomienia produkcji mikrokomputerów. Odpowiedzieli, że teraz potrzebują odżywek dla dzieci i narzędzi chirurgicznych, komputerami nie są zainteresowani. Nie rozumieją, że z eksportu mikrokomputerów byłyby i odżywki, i te narzędzia. Oni patrzą krótko — odżywka to odżywka, nic dalej.

µK: — To jest Pana najsilniejszy przeciwnik — okrzepła biurokracja.

L.W.: — To nawet nie to. Ludzie myślą zbyt krótkowzrocznie. Dzisiaj głównym problemem ekonomicznym jest nie zadłużenie czy marnotrawstwo, ale poziom umysłowy ludzi.

µK: — I to będzie Panu przeszkadzało.

L.W.: — Ale nie przeszkodzi! Hamulce trzeba osłabiać.

µK: — To są silne hamulce. Wiele dążeń już zatrzymały.

L.W.: — Życie bez problemów niewiele byłoby warte. Nie wystarczy głośno krzyczeć, trzeba jeszcze umieć przekonać. Każdy problem jest do rozwiązania, trzeba tylko znaleźć sposób. Zresztą, z ekonomicznego punktu widzenia, nawet tego krótkookresowego — mam rację.

µK: — Gdybyż rozstrzygnięcia ekonomiczne decydowały!

L.W.: — Tu zadecydują. To jest lawina, która już się toczy. Owszem, gdybym podjął się zawracania kijem rze-

Zwiastun

ki, to pewnie byłyby kłopoty. To byłoby wbrew naturze, prawom fizyki. Ale to właśnie moi przeciwnicy próbują to czynić. Człowiek jest na mikroinformatykę skazany, dlatego też ani mnie, ani nikomu innemu — nie przeszkodzą.

μK: — Co więc z tym krajem? Jaki będzie po rewolucji głoszonej przez Pana?

L.W. — Szalone upowszechnienie mikrokomputerów, gwałtowny rozwój edukacji — to zupełnie zmieni sytuację, przeobrazi ten kraj, nas wszystkich. Być może ominiemy okres fascynacji głupimi grami komputerowymi, właśnie kończący się na Zachodzie, od razu realizując program masowej mikroinformatycznej edukacji. Będziemy tworzyć za pomocą szarych komórek — surowca, którego nigdy nam nie braknie. Wystarczy tylko je rozwijać.

μK: — Czy spotyka się Pan ze zrozumieniem ze strony osób, które w ogóle zechciały Pana wysłuchać?

L.W.: — Ludzi na ogół nie interesuje szersze uzasadnienie. Myślą, jak zdobyć ciekawą grę, jak ją potem dobrze sprzedać, jak wykorzystać mikrokomputer za darmo, albo jak go z korzyścią wyprodukować. Dominuje krótkowzroczność, dążenie do zysku — za tydzień, za dwa.

μK: — I nie ma nikogo, kto dotrzymywałby Panu kroku?

L.W.: — Prawdę mówiąc — nie spotkałem. Ale to nic, niebawem się pojawią.

Opracował:
ZBIGNIEW GLUZA
Fot. TOMASZ MALANOWSKI

pasjonaci wszystkich pokoleń

μK

skupiajcie się!

wyprowadzony przez autonomiczną procedurę. Przerwy między zbiorami generowane są automatycznie do czasu zablokowania przerwań.

Odczyt

Program analizujący sygnał odczytu wykorzystuje ten sam programowany licznik, generujący przerwanie po 625 μs od chwili zainicjowania. Program wykorzystuje 4 bajty pamięci RAM.

Zmienna RDBT zawiera ostatnio rozpoznany bit — i przyjmuje wartość 80H dla bitu „1", 00H dla bitu „0" i 01H, jeżeli minął czas oczekiwania na zmianę sygnału wejściowego (700 μs po przerwaniu). Zmienna RDPR zawiera licznik rozpoznanych impulsów sygnału wejściowego. Licznik jest zerowany po stwierdzeniu, że nastąpiła zmiana częstotliwości (pięć jednakowych impulsów). Zmienna RDPS zawiera stan linii wejściowej przy poprzednim przerwaniu. Umożliwia to stwierdzenie, czy nastąpiła zmiana sygnału.

Zmienna RDSG służy do określenia czy następuje zmiana częstotliwości. Przyjmuje ona wartość —1 dla częstotliwości 1200 Hz i wartość 4 dla częstotliwości 600 Hz. W przypadku wykrycia impulsu o długości mniejszej od 625 μs zmienna RDSG jest pomniejszana aż do wartości —1. W przypadku wykrycia impulsu o długości większej od 625 μs, zmienna ta jest powiększana aż do wartości 4. Osiągnięcie skrajnej wartości oznacza zmianę bitu. Wynika z tego, że aby stwierdzona została zmiana, musi pojawić się pięć impulsów więcej o odmiennej częstotliwości od uprzednio rozpoznanej.

Algorytm programu obsługi przerwania dla odczytu przedstawia rysunek 2.

Odczyt danych z kasety magnetycznej sprowadza się do oczekiwania w pętli aż licznik RDPR (modulo 8) będzie równy 0 (osiem jednakowych próbek lub zmiana częstotliwości) oraz do odczytu w tych momentach zmiennej RDBT, a następnie złożenia z odczytanych bitów pełnej informacji. Należy jeszcze przeprowadzić analizę poprawności, polegającą na stwierdzeniu, czy:
— nie minął czas oczekiwania na kolejną próbkę (zmienna RDBT równa 01H)
— właściwe jest obramowanie bajtu (bit startu równy 0 i dwa bity stopu równe 1)
— odczytana suma kontrolna, zapisana z reguły za ostatnim bajtem, jest zgodna z sumą obliczoną podczas odczytu.

ADAM PLUTA
Przemysłowy Instytut Elektroniki
Warszawa

Dobre narzędzia to połowa sukcesu. Dlatego też poświęcamy im wiele miejsca. Opisywaliśmy już sposób programowej realizacji pracy krokowej, niebawem zamieścimy rozwiązanie sprzętowe... Nie jest bowiem prawdą (ach ci dziennikarze-popularyzatorzy o bezkresnej ignorancji), że programiście wystarczy papier i dobrze naostrzony ołówek. Papier i ołówek może posłużyć do napisania artykułu do mikroKLANU, program zaś — trzeba uruchomić!

W „skrzynce narzędziowej" programisty (programy: Monitor, DDT — ang. Designer Development Tool) najczęściej znaleźć można realizację tzw. chwilowych puntków wstrzymania. Spełniają one pokładane w nich nadzieje, dopóki nie przyjdzie nam testować programów iteracyjnych. Ustawianie pułapki przy każdym obrocie pętli może wzbogacić o niejeden siwy włos. Stąd koncepcja stałych pułapek.

Stałe pułapki

Realizacja nie kasujących się automatycznie pułapek (punktów wstrzymania) wymaga wykonania śledzonej instrukcji poza uruchamianym programem (np. w obszarze RAM zarezerwowanym na potrzeby systemu operacyjnego). Jedną z pojawiających się trudności jest konieczność zastąpienia instrukcji wywołania podprogramu (z instrukcjami RST włącznie) przez instrukcje skoku i odpowiednią korekcję stosu programu z uwzględnieniem różnych przypadków dla instrukcji warunkowych. Drugim problemem jest sprawa interpretacji instrukcji HLT i kodów nielegalnych.

W opisywanym rozwiązaniu punkty wstrzymania są realizowane przez umieszczenie pod wskazanymi adresami (muszą one odpowiadać adresom pierwszych bajtów instrukcji) kodu odpowiadającego RST 0. Wykonywane jest to bezpośrednio przed przekazaniem sterowania do uruchamianego programu, a powrót do programu zarządzającego (Monitor) powoduje zawsze odtworzenie stanu pamięci programu przez usunięcie wszystkich pułapek. Wybór instrukcji RST 0 nie jest krytyczny dla algorytmu obsługi pułapek i został podyktowany przyjętą konfiguracją pamięci systemu.

```
; Przekazywanie sterowania do programu
INSTR:  DS   1                    ; miejsce wykonywania instrukcji
        DS   1
        DS   1
        DS   1
;                                 pseudorejesty
PSAVE:  DS   2                    ; PC
SSAVE:  DS   2                    ; SP
```

```
ASAVE:  DS    2                    ; AF
HSAVE:  DS    2                    ; HL
DSAVE:  DS    2                    ; DE
BSAVE:  DS    2                    ; BC
;
;       sekwencja startu
;
RUN:    LXI   H, (NOP)+(NOP) SHL 8
        SHLD  INSTR
        MVI   H, (JMP (PSAVE))
        SHLD  INSTR+2
        LHLD  PSAV                 ; pobranie kodu
        MOV   C, M
        XCHG                       ; DE := adres
        CALL  LENGTH
        MOV   C, A                 ; długość instrukcji
        LXI   H, INSTR
COPY:   DCR   C                    ; wpisanie do INSTR
        JM    TEST
        LDAX  D
        MOV   M, A
        INX   D
        INX   H
        JMP   COPY
TEST:   MOV   A, B                 ; badanie kodu instrukcji
        CPI   (CALL 0)             ; jeżeli CALL. zastąpienie
        JZ    REPL                 ; przez JMP i korekta stosu
        ANI   (RST 0)
        CPI   (RST 0)              ; jeżeli RST wydzielenie
        JNZ   Ccnd?                ; adresu i dalej jak wyżej
        XRA   B
        MOV   L, A
        MVI   H, 0
        SHLD  INSTR+1
        JMP   REPL
Ccnd?:  CPI   (CNZ 0)              ; jeżeli wywołanie warunkowe,
        JNZ   GO                   ; to test na efektywność.
        XRA   B                    ; wydzielenie kodu warunku
        RAR
        RAR
        RAR
        RAR
        MOV   C, A                 ; 0-Z, 1-C, 2-P, 3-S
        SBB   A
        MOV   B, A                 ; rodzaj testu, 0 dla false
        LDA   ASAVE
        ANI   0C5H                 ; wydzielenie bitów warunkow.
        MOV   L, A
        RLC
        SBB   A
        MVI   H, 10H
        ANA   H
        ORA   L
SEL:    RRC
        RRC
        DCR   C
        JP    SEL
        ANA   H
        XRA   B
        ANA   H
        JNZ   GO                   ; skok przy nieefektywnym
REPL:   LHLD  SSAVE
        DCX   H                    ; korekcja stosu programu
        MOV   M, D
        DCX   H
        MOV   M, E
        SHLD  SSAVE
        MVI   A, (JMP 0)           ; JMP zamiast Ccnd
        STA   INSTR
GO:     XCHG                       ; adres po instrukcji
        SHLD  PSAVE
        ;
        ; ustawienie pułapek
        ;
        LXI   SP, SSAVE            ; odtworzenie rejestrów
        POP   H
        POP   PSW
        POP   D
        POP   D
        POP   B
        SPHL
        LHLD  HSAVE
        JMP   INSTR                ; i wreszcie START!
```

Przedstawiony algorytm, realizujący przekazanie sterowania do uruchamianego programu, wykorzystuje fragment pamięci wydzielony dla Monitora.

```
; Procedura LENGTH
; DZIAŁANIE: podaje ilość bajtów instrukcji, kodom nielegalnym i instrukcji
; HLT przypisuje zerową długość.
; WEJŚCIE: rej. C — kod instrukcji,
;          TABLE — tablica długości.
; WYJŚCIE: rej. A — ilość bajtów,
;          rej. B — kod instrukcji.
; ZMIENIA: B, C, H, L, A, F.
;
LENGTH: MOV   B, C                 ; przechowanie kodu
        MOV   A, C
        SUI   (HLT)
        RZ                         ; powrót z 0 dla HLT
        ADI   (HLT)+(MOV B, B)
        MOV   C, A                 ; C := kod + 40H
        SUI   (MOV B, B)+(MOV B, B)
        SBB   A
        INR   A                    ; 1 dla kodów 40-BF
        RNZ
        ADD   C                    ; A := C, CY:= 0
        RAR
        ORA   A
        RAR                        ; numer wiersza
        MOV   C, A
        MOV   A, B                 ; kod instrukcji
        MVI   B, 0
        LXI   H, TABLE             ; tablica długości
        DAD   B
        MOV   B, A                 ; kod instrukcji
        ANI   3
        MOV   C, A                 ; numer kolumny
        MOV   A, M
LNO:    RLC                        ; długość
        RLC                        ; do b1, b0
        DCR   C
        JP    LNO
        ANI   3
        RET
; Tablica długości przypisywanych kodom z zakresów
; OOH-3FH i OCOH-OFFH, zorganizowana w 32 wiersze
; po 4 kolumny. Element tablicy zajmuje 2 bity
;
TABLE:  DB    01011111B    ; RNZ  POP   JNZ   JMP
        DB    11011001B    ; CNZ  PUSH  ADI   RST
        DB    01011100B    ; RZ   RET   JZ
        DB    11111001B    ; CZ   CALL  ACI   RST
        DB    01011110B    ; RNC  POP   JNC   OUT
        DB    11011001B    ; CNC  PUSH  SUI   RST
        DB    01001110B    ; RC         JC    IN
        DB    11001001B    ; CC         SBI   RST
        DB    01011101B    ; RPO  POP   JPO   XTHL
        DB    11011001B    ; CPO  PUSH  ANI   RST
        DB    01011101B    ; RPE  PCHL  JPE   XCHG
        DB    11001001B    ; CPE        XRI   RST
        DB    01011101B    ; RP   POP   JP    DI
        DB    11011001B    ; CP   PUSH  ORI   RST
        DB    01011101B    ; RM   SPHL  JM    EI
        DB    11001001B    ; CM         CPI   RST
        DB    01110101B    ; NOP  LXI   STAX  INX
        DB    01011001B    ; INR  DCR   MVI   RLC
        DB    00010101B    ;      DAD   LDAX  DCX
        DB    01011001B    ; INR  DCR   MVI   RRC
        DB    00110101B    ;      LXI   STAX  INX
        DB    01011001B    ; INR  DCR   MVI   RAL
        DB    00010101B    ;      DAD   LDAX  DCX
        DB    01011001B    ; INR  DCR   MVI   RAR
        DB    00111101B    ;      LXI   SHLD  INX
        DB    01011001B    ; INR  DCR   MVI   DAA
        DB    00011101B    ;      DAD   LHLD  DCX
        DB    01011001B    ; INR  DCR   MVI   CMA
```

```
DB    00111101B   ;       LXI   STA   INX
DB    01011001B   ; INR   DCR   MVI   STC
DB    00011101B   ;       DAD   LDA   DCX
DB    01011001B   ; INR   DCR   MVI   CMC
```

Przypisanie instrukcji HLT i kodom nielegalnym długości równej zeru powoduje zaniechanie przepisywania ich do obszaru roboczego Monitora i przekazywanie sterowania bezpośrednio pod odpowiedni adres, uniemożliwiając ich realizację. Fakt ten można wykorzystać przy uruchamianiu programów, jeżeli na danym etapie pragniemy uniknąć wykonania określonej instrukcji. Wystarczy w tym celu chwilowo umieścić w odpowiednim miejscu programu jeden z ww. kodów i ustawić na nim pułapkę. Ten sam efekt można uzyskać umieszczając w programie instrukcję RST, wykorzystywaną do realizacji pułapek.

Poprawne działanie zaprezentowanego algorytmu wymaga nieprzerywalności realizacji całej sekwencji przekazywania sterowania. Rozwiązanie tego zagadnienia jest ściśle związane ze specyfiką architektury konkretnego systemu mikrokomputerowego wykorzystującego mikroprocesor 8080.

MICHAŁ BYSTRIANIN
Centrum Badań Kosmicznych
Warszawa

Zgodnie z zapowiedzią kontynuujemy kącik dla amatorów języka FORTH. Tym razem niezła porcja wiadomości. Ale, co najważniejsze, w oparciu o podany materiał możemy już tworzyć pierwsze programy! Zanim jednak FORTH stanie się przygodą, czeka nas jeszcze trochę żmudnego „przyswajania" podstawowych informacji. Staramy się skrócić do niezbędnego minimum część szkoleniową. Cierpliwości!

FORTH (2)

OPERACJE NA STOSIE

FORTH operuje na stosie danych (PARAMETER STACK) i stosie powrotów (RETURN STACK). Norma języka FORTH79 zawiera 12 słów do bezpośredniej obsługi stosów.

Przy opisie słów przyjęto następującą notację:
stan stosu danych : : =
(<dane na stosie przed wykonaniem słowa>
— <dane na stosie po wykonaniu słowa>)
stan stosu powrotów : : =
(<dane na stosie przed wykonaniem słowa>
— <dane na stosie po wykonaniu słowa>)
dane na stosie : : = $<n_k\ n_{k-1} \ldots n_2\ n_1>$
gdzie: n_1 liczba na szczycie stosu, n_k ostatnia liczba na spodzie stosu (lub ostatnia istotna dla danego słowa)

Powielenie liczby ze szczytu stosu **DUP**
DUP (<n>—<n n>)
Np. 5 DUP .. [CR] 5 5 OK

Poszczególne dyrektywy oddzielane są spacją. Każda kropka powoduje pobranie i wyświetlenie liczby ze szczytu stosu .[CR] to znak końca instrukcji: klawisz "new line")

Powielenie niezerowej liczby ze szczytu stosu **?DUP**
?DUP (<n>— { <n n> — gdy $n \neq 0$ | <n> — gdy $n = 0$ })
Np. 5 ?DUP .. [CR] 5 5 OK
0 ?DUP .. [CR] 0 EMPTY STACK (EMPTY STACK — komunikat o braku danych na stosie)

Zmniejszenie wskaźnika szczytu stosu **DROP**
DROP (<n>—<>)
Np. 3 4 DROP .. [CR] 3 OK

Zamiana dwóch danych ze szczytu stosu **SWAP**
SWAP ($<n_2\ n_1>$—$<n_1\ n_2>$)
Np. 1 5 SWAP .. [CR] 3 OK
(Zauważmy, że 1 5 .. [CR] 5 1 OK)

Skopiowanie na szczyt stosu drugiej od końca liczby na stosie **OVER**
OVER ($<n_2\ n_1>$—$<n_2\ n_1\ n_2>$)
Np. 2 3 OVER...[CR] 2 3 2 OK

Rotacja trzech pozycji ze szczytu stosu **ROT**
ROT ($<n_3\ n_2\ n_1>$—$<n_2\ n_1\ n_3>$)
Np. 3 2 1 ROT...[CR] 3 1 2 OK

Skopiowanie na szczyt stosu wybranej pozycji ze stosu **PICK**
PICK ($<n_k\ n_{k-1} \ldots n_2\ n_1\ i>$ — $<n_k\ n_{k-1} \ldots n_2\ n_1\ n_i>$)
Ostatnia wartość na stosie jest pobierana jako parametr określający położenie na stosie poszukiwanej liczby.
Np. 5 4 8 2 1 3 PICK.....[CR] 8 1 2 8 4 5 OK
Zwróćmy uwagę, że 2 PICK odpowiada OVER.

Rotacyjna zamiana miejscami wybranej liczby elementów ze stosu **ROLL**
ROLL ($<n_k \ldots n_1\ i>$ — $<n_k \ldots n_{i+1}\ n_{i-1} \ldots n_1\ n_i>$)
Ostatnia wartość na stosie jest pobierana jako parametr określający liczbę elementów podlegających rotacji.
Np. 1 2 3 4 5 6 4 ROLL.....[CR] 3 6 5 4 2 1 OK
Zwróćmy uwagę, że: 2 ROLL odpowiada SWAP
3 ROLL odpowiada ROT.

Badanie głębokości stosu **DEPTH**
DEPTH ($<n_k\ n_{k-1} \ldots n_2\ n_1>$ — $<n_k\ n_{k-1} \ldots n_2\ n_1\ k>$)
Wykonanie DEPTH powoduje odłożenie na szczycie stosu liczby odpowiadającej ilości danych na stosie w momencie wywołania słowa.
Np. 1 3 2 4 5 0 DEPTH.[CR] 6 OK

Przeniesienie ostatniej pozycji stosu danych na stos powrotów **> R**
> R (<n>—<>)[<>—<n>]

Przeniesienie ostatniej pozycji stosu powrotów na stos danych **R >**
R > (<>—<n>) [<n>—<>]
Np. 1 3 >R 12 15 R > ..[.. [CR] 3 15 12 1 OK

Skopiowanie ostatniej pozycji stosu powrotów na stos danych **R @**
R @ (<>—<n>) [<n>—<n>]

Np. 1983 > R 11 DUP 12 24 .. R @ ... R @... [CR] 24 12 1983 11 11 1983 OK

OPERACJE ARYTMETYCZNE

Dodawanie +
+ ($<n_2\ n_1>$—$<n_2+n_1>$)
Np. 2 4 +. [CR] 6 OK

Odejmowanie —
— ($<n_2\ n_1>$ — $<n_2 - n_1>$)
Np. 5 7 —. [CR] — 2 OK

Mnożenie *
* ($<n_2\ n_1>$ — $<n_1 * n_2>$)
Np. 2 3 *. [CR] 6 OK

Dzielenie /
/ ($<n_2\ n_1>$ — $<n_2/n_1>$)
Np. 14 7 /. [CR] 2 OK

A teraz kilka bardziej złożonych przykładów. Mamy zdefiniować słowa, które dają nastepujace wyniki:
SLOWO—X (<a b c d> — <((a+c)*b)/(c+d)>)
. SLOWO—X OVER + SWAP ROT >R ROT R> ROT ROT + * SWAP /;
SLOWO—Y (<a b c d e> — <((c + e) * (a + c))/((b + d) * (a + c)/e)>)
: SLOWO—Y > R > R ROT OVER + ROT R> + OVER * R> SWAP OVER / ROT ROT >R ROT R> + * SWAP /;
SLOWO—Z (<a b c> — <(((a + c) * b) + (c + b))/((a + c) * a)>)
: SLOWO—Z OVER OVER + >R ROT DUP >R + R> OVER * R> ROT >R ROT R> * + SWAP /;

Dla „treningu" proponujemy sprawdzić, jaki będzie efekt działania nastepujacego polecenia:
1 2 3 4 + ROT . OVER + . DUP + DUP DUP + .. [CR]

MAREK CZARZYŃSKI
ABAKUS

Tak się niefortunnie składa, że w chwili gdy przygotowujemy lutowe wydanie mikroKLANU, INFORMATYKA dociera do Czytelników z pewnym opóźnieniem. Być może dzięki naszym staraniom numer drugi faktycznie ukaże się w lutym, numer trzeci w marcu itd., zgodnie z kalendarzem. Na wszelki wypadek podajemy już teraz program obliczający datę Wielkanocy.

W połączeniu z algorytmami „kalendarzowymi" (INFORMATYKA nr 9/83) można wzbogacić mikrokomputer w kalendarz, co pozwoli na prowadzenie harmonogramu zajęć, spotkań, spraw do załatwienia, terminów etc.

Procedura „Wielkanoc"

```
10 REM PROGRAM PODAJE DATE WIELKANOCY WG KALENDARZA
20 REM GREGORIANSKIEGO CZYLI OD ROKU 1583 WŁACZNIE.
30 REM DLA DAT WCZESNIEJSZYCH (KALENDARZ JULIANSKI)
40 REM NALEŻY PODSTAWIC:
50 REM M = 15 ORAZ P = 6 (PATRZ LINIE 130 oraz 150).
60 PRINT "ROK?";
70 INPUT Y
80 LET A = Y—INT(Y/19)*19
90 LET B = Y—INT(Y/4)*4
100 LET C = Y—INT(Y/7)*7
110 LET T = INT(Y/100)
120 LET P = 4+T—INT(T/4)
130 LET M = P—INT(P/7)*7
140 LET P = 15+T—INT(T/4)—INT((8*T+13)/25)
150 LET P = P—INT(P/30)*30
160 LET P = 19*A+P
170 LET D = P—INT(P/30)*30
180 IF D < 28 THEN 220
190 IF D = 29 THEN 210
200 IF A < = 10 THEN 220
210 LET D = D—1
220 LET P = 2*B+4*C—D+M
230 LET P = P—INT(P/7)*7
240 LET D =D+P
250 IF D > 9 THEN 290
260 LET D = D+22
270 PRINT "WIELKANOC: " ;D; "MARCA"; Y
280 STOP
290 LET D = D—9
300 PRINT "WIELKANOC:";D;"KWIETNIA"; Y
310 END
```

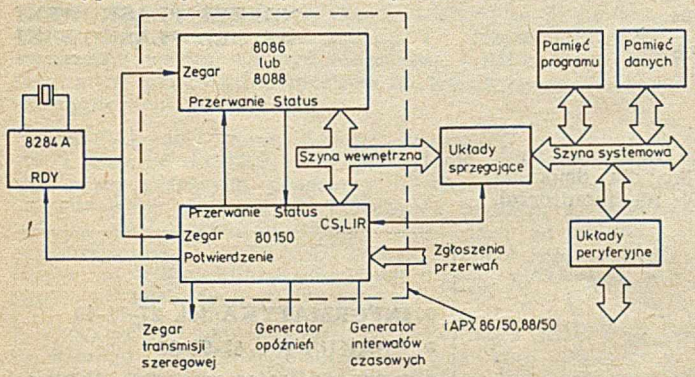
Przykłady pracy programu:

```
ROK? 1900
WIELKANOC: 15 KWIETNIA 1900
ROK? 1921
WIELKANOC: 27 MARCA 1921
ROK? 1984
WIELKANOC: 22 KWIETNIA 1984
ROK? 2000
WIELKANOC: 23 KWIETNIA 2000
ROK? 4356
WIELKANOC: 1 KWIETNIA 4356
```

Opracował
JAKUB TATARKIEWICZ
na podstawie „PPC Calculator Journal", V1ON8P4.

Krzemowy CP/M

Popularny system operacyjny CP/M przechowywany jest zazwyczaj na dyskietce. Niedawno firma INTEL wprowadziła na rynek specjalny układ VLSI 80150, który realizuje funkcje CP/M 86 (udoskonalona wersja CP/M 80 dla systemów iAPX86, iAPX88 oraz iAPX186). Przeznaczony jest on do zastosowania w mikrokomputerach i umożliwia korzystanie z systemu operacyjnego nawet w urządzeniach nie wyposażonych w dyski elastyczne.



Schemat blokowy systemu mikroprocesorowego wykorzystującego układ 80150 przedstawiono na rysunku. Zastosowanie 80150 jest najbardziej ekonomiczną formą realizacji CP/M 86 w mikrokomputerze. Ponieważ wszystkie funkcje sprzętowe wymagane przez system operacyjny zostały zrealizowane w jednym układzie scalonym, oszczędza się na kosztach związanych z projektowaniem mikrokomputera. Zmniejszają się koszty montażu i testowania urządzeń. Wzrasta równocześnie niezawodność systemu, gdyż programy — w przypadku błędu — nie mogą zniszczyć oprogramowania systemowego zawartego w pamięci RAM.

Jeżeli system nie jest wyposażony w stację dysków elastycznych, można ustalić odpowiedni blok pamięci RAM, który będzie traktowany przez system operacyjny jak dyskietka. Realizowanie takich funkcji, jak otwieranie i zamykanie zbiorów, przechowywanie programów itp. — odbywa się analogicznie jak przy stosowaniu dyskietek; z tą tylko różnicą, że wzrasta efektywność systemu, jako że wszystkie operacje wykonywane są w RAM. Programy użytkowe mogą być ładowane przy użyciu konwencjonalnych technik sprzęgających (z pamięci kasetowej, domenowej, stałej) lub nawet z klawiatury.

Opracował AJP
na podstawie
„SOLUTIONS" I/II 1983

- Istnieje możliwość uzyskania (w drodze wymiany) programów dla ZX SPECTRUM. Piotr Parlewicz (tel. w Warszawie: 42-40-94) posiada już zbiór ponad 80 gier i programów użytkowych. Wśród tych ostatnich warto wymienić: **PASCAL, FORTH, ASSEMBLER/DISASSEMBLER/EDITOR, Slow Loader** (ładowanie programów z ZX81), **VU-File** (baza danych).
- Wydany kilka miesięcy temu w niewielkim nakładzie, ale za to na kredowym papierze, **„Katalog Skrócony CEMI 82/83"** (cena 400 zł) jest już niestety dostępny tylko na czarnym rynku. Za ten skrót ze skrótów niektórzy płacą 3 tys. zł!
- BOINTE przy Przemysłowym Instytucie Elektroniki (Warszawa, Marynarska 10, tel. 47-07-62) wydaje zeszyty **„Systemów Mikroprocesorowych"**, w cenie 345 zł za egzemplarz (wyjątkowo drogo!). Można w nich znaleźć opisy profesjonalnych urządzeń lub rozwiązań wykorzystujących układy mikroprocesorowe. Po dłuższej przerwie ukazały się ostatnio zeszyty 11, 12 i 13.
- Przypominamy. Trwa **konkurs na program gry edukacyjnej** organizowany przez klub ABAKUS. Do wygrania m.in. ZX SPECTRUM 48 K (cena „wolno-rynkowa" — ok. 160 tys. zł).
- **Klub Użytkowników Mikroprocesorów (KUM)** zajmuje się wymianą informacji i usług. Ankietę klubową można otrzymać od założyciela klubu — Andrzeja Drożniaka; ul. Pytlasińskiego 14 m 4, 00-707 Warszawa; tel. dom. 41-26-01.

U nas to kawał żmudnej i niewdzięcznej roboty... W krajach rozwiniętych — projektowanie precyzyjnych detali mechanicznych, płytek drukowanych, struktur układów scalonych — to czynności wykonywane przy użyciu mikrokomputerów. Rzecz nie tylko w przeprowadzeniu obliczeń — efektem końcowym jest precyzyjny rysunek (niezależnie od innych nośników informacji wykorzystywanych potem do sterowania urządzeniami technologicznymi).

Do niedawna dla utrwalenia wyników pracy na papierze stosowany był ploter. Od kilku lat ogromnie popularne stało się wykorzystywanie do tych celów drukarek mozaikowych, które oprócz liter mogą stawiać punkty układające się w obraz. Na Targach Poznańskich zaprezentowano drukarkę D-200 (polska!), którą można wykorzystać do celów graficznych. Lecz jak to często bywa z prezentowanymi tam wyrobami — drukarka taka nie jest sprzedawana na krajowym rynku. Dlatego też poniższy przykład opracowany został na komputerze BBC (patrz rozmowa w poprzednim wydaniu mikroKLANU) współpracującym z drukarką SEIKOSHA. Warto zwrócić uwagę, że banalne w sumie oprogramowanie w połączeniu z wcale nie tak znów wyrafinowanym sprzętem — daje rewelacyjne wyniki! No cóż, wydawałoby się, że możemy tylko pomarzyć... Tymczasem grupa zapaleńców już postanowiła przerobić poczciwą DZM 180, tak aby uzyskać wyniki nie gorsze od prezentowanych w poniższym tekście. W jednym z najbliższych wydań mikroKLANU opublikujemy ich doświadczenia.

Kopiowanie ekranu na drukarkę graficzną

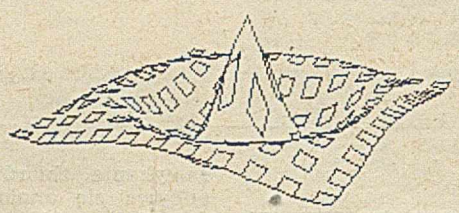
Jeżeli nie dysponujemy ploterem można napisać program kopiujący obraz z monitora CRT na drukarkę mozaikową. W oprogramowaniu mikrokomputera BBC — użytego w przedstawionym rozwiązaniu — znajduje się funkcja POINT (X, Y), która analizuje punkt o współrzędnych X, Y na ekranie monitora. Przyjmuje ona wartość odpowiadającą kolorowi danego punktu.

Jeżeli — na przykład — wykorzystywane są tylko dwa kolory, funkcja POINT przyjmuje wartość 0 dla punktu czarnego, a dla punktu białego — wartość 1.

Maksymalna rozdzielczość monitora ekranowego wynosi 1280 × 1024 punkty. W zależności od trybu pracy, rozdzielczość monitora można zmniejszyć (na plamę świetlną składa się kilka lub kilkanaście punktów):
tryb 0 — 640 × 256
tryby 1, 4 — 320 × 256
tryby 2, 5 — 160 × 256
tryby 3, 6, 7 — nie dają możliwości graficznych.

W trybie 4, dla którego opracowano poniższy przykład, punkt świetlny ma rozmiar 4 × 4, co należało uwzględnić przy pisaniu programu.

W drukarce znakowo-mozaikowej SEIKOSHA, za pomocą której kopiowano obraz z monitora, znaki tworzone są z matrycy punktów o wymiarach 6 × 7 (6 kolumn po 7 punktów każda). Drukowanie graficzne jest realizowane w sposób następujący. Do drukarki należy przesłać kod drukowania graficznego VDU 1,8, a następnie ciąg liczb. Każda liczba jest kodem jednej kolumny zawierającej punkty odbijane przez siedem igieł. Liczba ta musi być zawarta w przedziale od 128 do 128 + 2^7 (liczby od 0 do 127 zarezerwowane są dla kodów ASCII).

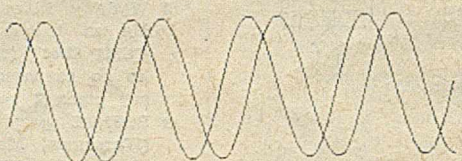


Rys. 1. Obraz graficzny funkcji Z = cos (a/1,5) · EXP (—a/5) · 5, gdzie $a = \sqrt{x \cdot x + y \cdot y}$ – zrealizowany na drukarce mozaikowej

Oto przykład określania kodu:

```
o
•
•
o A = 128+(2¹+2²+2⁴+2⁶)
•
o
•
```

VDU 1 oznacza, że następny znak przeznaczony jest tylko dla drukarki (nie jest wyświetlany na monitorze).



Rys. 2. Wykresy funkcji y = sin x i y = = cos x — zrealizowane na drukarce mozaikowej

Jeżeli zmiennym A, B, C zostaną przypisane żądane kody, zapis instrukcji drukowania może wyglądać następująco:
VDU 1, 8, A, B, C,...
lub też
VDU 1, 8
VDU 1, A
VDU 1, B
VDU 1, C

Włączenie drukarki następuje przez VDU 2, a wyłączenie drukarki przez VDU 3.

Dla kodów ASCII drukarka drukuje do 80 znaków w rzędzie, tak więc maksymalna rozdzielczość w rzędzie wynosi 480 punktów (80 × 6).

Procedura kopiowania została napisana w języku BASIC:

```
10 DEFPROCHARTCOPY
20 VDU2, 1,8:FORY% = 812T00STEP—28:
                                VDU13
30 FORX% = 0T01200STEP4:A% = 128
40   FORN% = 0T027STEP4
50     IF POINT(X%,   Y%—N%) = 1   GO-
                                SUB90
60     NEXT : VDU1,A% : NEXT
70 NEXT. : VDZ13,3,4,1,15
80 ENDPROC
90 ONN%DIV4 + 1GOTO100,110,120,130,140,150,
                                160
100 A% = A%+1:RETURN
110 A% = A%+2:RETURN
120 A% = A%+4:RETURN
130 A% = A%+8:RETURN
140 A% = A%+16:RETURN
150 A% = A%+32:RETURN
160 A% = A%+64:RETURN
```

Za pomocą powyższego programu skopiowano wykresy funkcji przedstawionych na rysunkach.

Czas kopiowania monitora
— przy całym obrazie z monitora ok. 15 min
— przy jednym wykresie ok. 4 min
— przy kilku (rzędu 10) ok. 6 min.

Istnieje również możliwość drukowania negatywu obrazu z monitora. Należy wówczas zmienić w wierszu 50 POINT (X%, Y%—N%) = 1 na POINT (X%, Y%—N%) = 0.

ANDRZEJ PLĄSKOWSKI
WITOLD PLĄSKOWSKI
Warszawa

- **INFORMATYKA tel. 27-71-40**
- **ABAKUS tel. 42-91-85**
- **KUM tel. 41-26-01**
- ?

— prowadzi Andrzej J. Piotrowski (tel. dom. 48-22-85)

Ciąg dalszy ze str. 12

ALGOL 68 ma dwa sposoby gospodarki pamięcią: lokalny i stertowy. Lokalny odpowiada powszechnie stosowanej gospodarce blokami, która zwykle realizowana jest za pomocą stosu o organizacji LIFO; sterta natomiast jest to spójny obszar pamięci, w którym generatory (stertowe) mogą rezerwować pamięć. W przypadku gospodarki lokalnej, w momencie wchodzenia do bloku — zostaje mu przydzielona pamięć lokalna, która jest zwalniana po wyjściu z bloku. Tymczasem obiekty stertowe, umiejscowione w czasie działania bloku, po wyjściu z niego nie giną.

W przeciwieństwie do PL/I, ALGOL 68 dopuszcza jedynie niewielką liczbę automatycznych konwersji, natomiast w przeciwieństwie do ALGOLU 60 — zawiera pewną liczbę procedur wejścia i wyjścia. W ALGOLU 68 istnieją też pewne narzędzia do programowania równoległego.

Implementacje ALGOLU 68 zostały zrealizowane przez kilka różnych grup na różnych maszynach.

PASCAL

PASCAL jest wg autorki najważniejszym z języków, jakie powstały po PL/I i są obecnie używane. Opracował go Niklaus Wirth w 1968 roku, a pierwszy działający kompilator powstał w 1970 roku. Ostateczny opis języka opracowali w 1978 roku Jensen i Wirth. Jest to jedyny, szeroko stosowany język, nad którym początkowo pracowała tylko jedna osoba. Drugim takim językiem był opracowany przez Kenneth Iversona — APL.

PASCAL jest językiem o strukturze blokowej, a jego nowatorstwo polegało na opracowaniu możliwości definiowania nowych typów danych. PASCAL wprowadził koncepcję typów wyliczeniowych, która polega na tym, że dla danej zmiennej definiowane są uporządkowane, skończone zbiory wartości; ponadto na ich podstawie można potem definiować tzw. typy okrojone. Przykładowo — użytkownik może napisać:

```
type dzień = (pon,wt,śr,czw,pt,sob,niedz);
oraz
type roboczydzień = pon..pt;
```

Język dopuszcza regularne tablice oraz struktury złożone z niejednorodnych elementów zwanych rekordami. Poza tym różne zmienne tego samego typu mogą mieć różniące się struktury. Sytuacja ta występuje wtedy, gdy w opisie danego typu podano kilka dopuszczalnych wariantów.

We wspomnianym już raporcie Jensen i Wirth stwierdzili: „Opracowanie języka PASCAL opierało się na dwóch głównych założeniach. Po pierwsze miał to być język odpowiedni do nauczania programowania, traktowanego jako systematyczna dziedzina wiedzy, oparta na pewnych fundamentalnych założeniach, które znalazłyby jasne i naturalne odzwierciedlenie w języku. Po drugie zakładano, że realizacja języka ma być efektywna i niezawodna, na tyle, na ile jest to tylko możliwe na współczesnie dostępnych komputerach".

PASCAL jest coraz częściej nauczany na uniwersytetach, chociaż do 1980 roku nie był on zbyt często używany przez piszących programy użytkowe. PASCAL został zrealizowany dla dużych i średnich maszyn, a także dla minikomputerów. Kompilatory PASCALA są zwykle pisane również w PASCALU. Znane są także jego implementacje na kilku mikrokomputerach oraz niektórych komputerach osobistych.

Powszechnie używany jest „P code" jako język pośredni, z którego następnie generowany jest kod docelowy dla różnych komputerów. PASCAL jest wykorzystywany jako punkt wyjścia przy opracowywaniu nowych języków (np. języka ADA), tak jak poprzednio ALGOL 60 był punktem wyjścia dla PASCALA i innych języków.

Chociaż nowatorstwo PASCALA polegało na wprowadzaniu pewnych nowych rozwiązań dotyczących typów danych, to jednak główną zasługą jego twórców był sposób wyboru i połączenia cech innych języków. Warto przypomnieć, że PASCAL zawiera również kontrowersyjną instrukcję GOTO.

Brinch Hanson dodając w 1977 do tego języka kilka nowych elementów, opracował CONCURRENT PASCAL, przeznaczony do programowania współbieżnego. Również Wirth wykorzystał język PASCAL do opracowania w 1977 r. MODULI, szczególnie przydatnej do wieloprogramowania i programowania w czasie rzeczywistym.

CLU

Liskov i jego współpracownicy stwierdzili w charakterystyce tego języka: „CLU został zaprojektowany, aby umożliwić zastosowanie się do metodologii, która zakłada opracowywanie programów oparte na dekompozycji problemu oraz na wyróżnianiu pewnych pojęć abstrakcyjnych (ang. abstractions)" oraz „CLU jest nowym językiem programowania, opracowanym po to, aby w trakcie konstruowania programu umożliwić korzystanie z pewnych pojęć abstrakcyjnych.

Prace w zakresie metodologii programowania doprowadziły do stwierdzenia, że w procesie programowania pomagają następujące trzy rodzaje wyabstrahowanych narzędzi: pojęcie abstrakcyjnej proceduralności (ang. procedural abstraction), pojęcie abstrakcyjnego sterowania (ang. control abstraction), a zwłaszcza pojęcie abstrakcyjnych danych (ang. data abstraction). Z powyższych trzech pojęć tylko pojęcie abstrakcyjnej proceduralności jest dobrze rozwiązywane przez konwencjonalne języki — za pomocą procedur lub podprogramów. Natomiast w CLU są poza tym, także nowe mechanizmy lingwistyczne, które zapewniają uwzględnienie pojęć abstrakcyjnych danych i abstrakcyjnego sterowania".

Program w języku CLU składa się z grup modułów. CLU dopuszcza trzy rodzaje modułów: procedury umożliwiające abstrakcyjne ujęcie proceduralności, iteratory zapewniające możliwość realizacji pojęcia abstrakcyjnego sterowania oraz wiązki ułatwiające abstrakcyjne podejście do danych.

Procedury mogą być bezparametrowe lub mieć kilka argumentów. Mogą one także, choć nie muszą, dostarczać pewną liczbę wyników.

Iterator jest wywoływany przez instrukcję **for** zawierającą **ciało instrukcji**, które ma być wykonane jeden raz dla każdej danej.

Wiązka natomiast pozwala tworzyć typ danych, który jest zbiorem obiektów oraz zbiorem operacji. Operacje mogą być alko procedurami, albo iteratorami. Obiekty danego typu mogą być tworzone i wykorzystywane tylko za pomocą operacji dotyczących tego typu i dzięki temu informacja reprezentująca obiekt jest całkowicie zasłonięta przez wiązkę.

Procedury, iteratory i wiązki mogą mieć parametry umożliwiające, w ramach jednego modułu, definicję klasy, powiązanych między sobą abstrakcyjnych pojęć. Przykładowo, jedna wiązka może definiować zbiory o określonym typie elementu. Podstawowymi elementami CLU są obiekty i zmienne. Obiekty są to dane, na których działają programy; zmienne określają nazwy używane do odwoływania się do obiektów. Do danego obiektu można odwoływać się z innych obiektów, ale także z tego samego. Dzięki temu, chociaż nie ma typów explicite opisujących odwołania, możliwe są rekurencyjne i wspólnie użytkowane struktury danych.

CLU został zrealizowany w MIT i działa na komputerze DEC System 20.

EUCLID

Jednym z interesujących eksperymentów w dziedzinie języków programowania było zdefiniowanie języka EUCLID przez B. W. Lampsona, J. J. Horninga, R. L. Londona, J. G. Mitchella i G. L. Popeka. Znajdowali się oni w różnych miejscach, pracując dla różnych organizacji, a komunikowali się przede wszystkim za pośrednictwem sieci komputerowej ARPANET (Advanced Research Projects Agency Network). Pierwszy opublikowany dokument, zawierający definicje języka, opracował Lampson ze współpracownikami w 1977 r. Głównym celem twórców EUCLIDA było stworzenie nowego języka programowania przeznaczonego do pisania programów systemowych, które mogłyby być weryfikowane. EUCLID jest pewną modyfikacją PASCALA, a nawet można go uważać za nową wersję PASCALA, uwzględniającą konieczność weryfikacji programów. Nawet raport na temat języka EUCLID jest znacznie przeredagowaną wersją poprawionego raportu PASCALA. Twórcy języka przyznają, że wykorzystywali także rozwiązania i niektóre cechy innych języków.

W raporcie EUCLIDA wskazano na następujące różnice w porównaniu z PASCALEM:

• program w języku EUCLID musi wymieniać wszystkie identyfikatory używane w modułach lub poza nimi

• dwa identyfikatory, które mają taki sam zsięg, nie mogą odnosić się do tych samych lub nakładających się zmiennych

• program może kontrolować formalnie przydział pamięci dla dynamicznych zmiennych

• wprowadzono nowy rodzaj rekordu, tzw. moduł (ang. module), zawierający podprogramy i typy, który ułatwia modularyzację

• można korzystać w kontrolowany sposób z właściwości maszyny i ewentualnie rezygnować ze sprawdzania typów;

• język umożliwia badanie asercji

• pominięto pewne elementy języka PASCAL, a w szczególności wejścia-wyjścia, rzeczywiste, wielowymiarowe tablice, etykiety, skoki, funkcje i procedury jako parametry.

Implementację EUCLIDA wykonano na Uniwersytecie w Toronto na minikomputerze PDP-11.

SETL

Język SETL opracowano na Uniwersytecie New York w Courant Institute. Podstawową koncepcją i konstrukcją semantyczną tego języka jest zbiór matematyczny. W języku SETL opracowano proste operacje ułatwiające m.in. sprawdzanie przynależności do zbiorów, znajdowanie sumy i przecięcia zbiorów a także liczenie mocy zbiorów.

SETL zawiera ogólne typy danych zarówno dla zbiorów nieuporządkowanych, jak i dla ciągów, a ponadto umożliwia korzystanie z wyrażeń logicznych oraz kwantyfikatorów ogólnych rachunku predykatów I rzędu.

SETL tym różni się od wielu eksperymentalnych języków, że nie ma silnej specyfikacji typów. Jednym z celów, jakie stawiali sobie twórcy tego języka, było dopuszczenie możliwości wyrażania algorytmu w taki sposób, aby był on niezależny od szczególnego wyboru struktury danych. Programy napisane w SETL mają bardzo niewiele deklaracji.

Najważniejszą właściwością, która w pewnym stopniu dowodzi, że jest to język „bardzo wysokiego poziomu", jest sposób pisania algorytmów. Tworzy się je na bardzo wysokim poziomie abstrakcji, a dopiero potem doskonali pod względem wydajności.

SETL może być przykładem programowania zstępującego, w którym można wykonywać już początkowe jego fazy. Dodatkowo na niższych poziomach można także dostarczać informacje o strukturach danych.

Implementacja SETL została zrealizowana na Uniwersytecie New York na komputerach CDC 6600, PDP-10, VAX 11/780 oraz IBM System/370.

ADA

W ciągu kilku ostatnich lat można było spotkać się z terminami: DOD-1, STRAWMAN, TINMAN, IRONMAN, STEELMAN, ADA, a także BLUE, GREEN, RED, YELLOW. Terminy te odnoszą się do pracy podjętej w 1975 r. przez Departament Obrony USA (DOD). Ze względu na ogromne znaczenie tego projektu nie tylko dla tej instytucji, ale i dla rozwoju całej informatyki, zostaną tu omówione: tło, przyczyny powstania projektu oraz sposób jego prowadzenia.

W 1975 r. pracownicy DOD doszli do wniosku, że liczba używanych języków wysokiego poziomu staje się zbyt duża. Aby przeciwdziałać temu zjawisku wydano w 1976 r. dwa zarządzenia. Pierwsze z nich mówiło m.in., że „Do oprogramowania systemów służących obronie narodowej mogą być stosowane tylko takie języki wysokiego poziomu, które zostały zatwierdzone przez DOD, z wyjątkiem przypadków, gdy okaże się, że żaden z tych języków nie jest wystarczająco ekonomiczny albo nie jest odpowiednio przydatny dla danego problemu (...). Każdy z zatwierdzonych języków będzie nadzorowany przez odpowiednią instytucję."

Duża różnorodność języków oraz ciągłe wydatki na ich rozszerzanie i konserwację sprawiły, że zaczęto szukać mniej kosztownych rozwiązań. W tym celu powołano Grupę Roboczą ds. Języków Wysokiego Poziomu (HOLWG — High-Order Language Working Group), która miała:

— sformułować warunki, jakim powinien odpowiadać język wysokiego poziomu powszechnie stosowany w DOD
— porównać te warunki z cechami istniejących języków
— zaproponować adaptację lub implementację wybranego języka.

Analizy zastosowań przeprowadzone przez HOLWG w trzech gałęziach sił zbrojnych (armia lądowa, lotnictwo i marynarka) wykazały, że mogłyby one posługiwać się jednym, wspólnym dla wszystkich, językiem wysokiego poziomu, co nie było wcale oczywiste przed rozpoczęciem badań. Początkowo język roboczo nazwano DOD-1, a następnie w 1978 r. oficjalnie — ADA (na cześć Ady Augusty Lovelace, asystentki Charlesa Babbagge, będącej przypuszczalnie pierwszym w historii programistą). Jako dziedziny zastosowań tego języka przewidywano wbudowane systemy komputerowe, wchodzące w skład dużych systemów wojskowych, gdzie oprogramowanie jest ich integralną częścią.

Dokumenty określające wstępne wymagania i umowy na projekt języka

Prace rozpoczęto od opracowania dokumentu o kryptonimie STRAWMAN, który szczegółowo określał warunki, jakim powinien odpowiadać nowy język: zmniejszenie kosztów oprogramowania, przenośność, łatwość konserwacji, wysoka niezawodność, czytelność, dostatecznie efektywny kod wynikowy.

Warunki STRAWMAN zostały rozesłane do różnych departamentów wojskowych, urzędów państwowych, ośrodków akademickich i przemysłowych w celu uzyskania wstępnej oceny. Wiele z uzyskanych odpowiedzi zostało opublikowanych w sieci komputerowej ARPANET. Opierając się na uzyskanych opiniach, opracowano dwa dalsze dokumenty z założeniami: WOODENMAN i TINMAN. W 1976 r. przeanalizowano, pod kątem wymagań stawianych przez TINMAN, ponad 20 języków wysokiego poziomu. Brano także pod uwagę możliwość adaptacji jednego z nich, jeśli konieczne zmiany nie byłyby zbyt duże. Nie znaleziono jednak takiego języka. Stwierdzono natomiast, że PL/I, ALGOL 68 i PASCAL, są językami o tak wielkich możliwościach, że każdy z nich mógłby być wybrany jako podstawa do budowy potrzebnego języka.

W 1977 r. DOD wydał następny zbiór założeń o nazwie IRONMAN, a w czerwcu tego roku zawarto cztery umowy na opracowanie konkurencyjnych, wstępnych projektów języka (faza nr 1). Projekty te, dla zachowania anonimowości nazwane kolorami, przedstawiono DARPA (Defense Advanced Projects Research Agency) w lutym 1978. W ciągu następnego miesiąca w publicznej dyskusji nad projektami mogli brać udział wszyscy, którzy mieli dość czasu, aby przeczytać obszerną dokumentację — wypowiadały się osoby indywidualne i grupy ze środowisk uniwersyteckich, wojskowych i przemysłowych, nie tylko z USA, ale także z Europy. W kwietniu 1978 zawarto dwie umowy na wykonanie fazy nr 2 projektu: z firmą INTERMETRICS (projekt RED) oraz z CII HONEYWELL-BULL (projekt GREEN).

Podstawy techniczne języka

W fazie nr 1 wszyscy projektujący wybrali PASCAL jako język bazowy, chociaż nie należało to do wymagań TINMAN. Stąd, oczywiście ostateczna wersja języka ADA jest oparta na PASCALU, choć istnieją nieporozumienia dotyczące tego określenia. Wielu ludzi sądzi, że ADA jest tylko rozszerzeniem PASCALA, natomiast stwierdzenie „ADA jest oparta na PASCALU" oznacza, że wykorzystano koncepcje i cechy PASCALA, nie wprowadzając niepotrzebnych zmian syntaktycznych (oczywiście „nie potrzebny" dla każdego oznacza co innego).

Proces ten jest analogiczny do tworzenia PL/I, o którym można powiedzieć, że jest oparty na FORTRANIE lub ALGOLU. W PL/I istnieją wprawdzie elementy pochodzące z tych języków, jak również z COBOLU i innych języków, nie można jednak w żadnym razie uważać go tylko za proste rozszerzenie któregoś z nich.

Przykłady i ogólny podział wymagań

Przykład ADY jest, jak dotąd, jedyną próbą racjonalnego tworzenia założeń języka, podjętą na tak dużą skalę przed przystąpieniem do opracowania projektu. Warto

przytoczyć dwa fragmenty z dokumentu DOD (1978). Pierwszy z nich dotyczy liczb całkowitych i stałoprzecinkowych, drugi — sterowania sekwencyjnego:

• „Liczby całkowite i stałoprzecinkowe należy traktować jako dokładne wartości numeryczne. Nie wykonuje się ukrytego zaokrąglania lub obcinania w działaniach na liczbach całkowitych i stałoprzecinkowych".

• „Potrzebny jest mechanizm sterujący sekwencyjnym wykonywaniem instrukcji. Język nie powinien wprowadzać ograniczeń stylu programowania, takich jak wybór między ogranicznikami a separatorami instrukcji, chyba że takie ograniczenia zmniejszyłyby liczbę błędów podczas programowania".

Czytelnicy znający opisy języków wysokiego poziomu wiedzą, że są to wymagania, a nie szczegółowy opis języka. Ostateczna wersja wymagań — nazwana STEELMAN — została opracowana w czerwcu 1978. Korzystały z niej oba wybrane zespoły opracowujące ostateczną wersję języka.

Wymagania te można podzielić według następujących tematów: ogólne kryteria projektu, ogólna składnia, typy (np. liczbowe, wyliczeniowe i złożone, zbiory), wyrażenia stałe, zmienne i ich zakresy, klasyczne mechanizmy sterujące, funkcje i procedury, wejście-wyjście, formatowanie i kontrola konfiguracji maszyny, przetwarzanie współbieżne, obsługa sytuacji wyjątkowych, reprezentacja i inne narzędzia ułatwiające translację, translacja i biblioteki, otoczenie języka.

Wymagania związane z otoczeniem języka

Równolegle z uzupełnieniem wymagań od 1978 r. prowadzono prace nad ustaleniem warunków, jakim powinno odpowiadać środowisko, w którym — po zaprojektowaniu i implementacji — ma być wykorzystywany język. Środowisko — to takie problemy, jak: wybór osoby decydującej o ewentualnych zmianach i poprawkach w języku, sposob walidacji kompilatorów, metody szkolenia użytkowników, sposób rozwiązania biblioteki zastosowań oraz edytorów itp. W lutym 1980 wydano dokument STONEMAN omawiający to środowisko, który był również raczej listą wymagań, a nie gotowym planem implementacji języka.

Faza testów i ocen

Od kwietnia 1979, kiedy wybrano ostatecznie wersję GREEN (firmy CII HONEYWELL-BULL), aż do grudnia 1979 — DOD dokonywał testów i oceny działania języka. Jak poprzednio, zwrócono się do środowisk informatycznych z prośbą o przekazywanie uwag. Faza ta polegała przede wszystkim na pisaniu próbnych programów dla istniejących zastosowań. W ten sposób sprawdzono rzeczywiste możliwości wykorzystywania ADY oraz przygotowywano propozycje ewentualnych ulepszeń języka. W akcji tej wzięło udział ponad 45 organizacji, w których napisano ok. 90 programów (w tym kilka dużych).

Kolejne wersje projektu

W maju 1979 HOLWG wybrała, jako ostateczny, projekt opracowany we Francji przez CII HONEYWELL-BULL pod kierunkiem Jean Ichbiah. Już w czerwcowym numerze ACM SIGPLAN NOTICES opublikowano definicję języka. Przyczyniło się to do szerokiej popularyzacji ADY, gdyż sam SIGPLAN liczył wówczas ponad 9000 członków, a ponadto DOD dysponował listą kilku tysięcy potencjalnych ekspertów. Oczekiwano na opinie dotyczące zarówno samego języka, jak i wyniki testów. W lipcu 1979 powołano grupę około 10 wybitnych specjalistów celem przeanalizowania nadesłanych opinii i przekazania ich wraz z oceną do zespołu autorskiego. W lipcu 1980 zespół Ichbiaha ukończył ostateczną wersję projektu języka, który został rozesłany do opiniodawców, przewidzianych jako uczestników procesu standaryzacji ADY w ANSI metodą dyskusji.

Główne cechy języka

Ze względu na złożoność i szerokie możliwości trudno jest scharakteryzować ADĘ w sposób zwięzły. Autorka ograniczyła się więc do pokazania tylko niektórych, szczególnie atrakcyjnych wg niej cech języka.

ADA zawiera silny mechanizm pozwalający użytkownikowi na określanie rozmaitych typów danych, a także ich ograniczanie poprzez podawanie podzakresów, podtypów i typów pochodnych. Szczególną uwagę zwrócono — ze względu na potencjalne zastosowania — na obiekty typu stałoprzecinkowego. Reguły dotyczące nazewnictwa i tworzenia wyrażeń są podobne do istniejących w innych językach, choć nie wolno łączyć w wyrażeniach obiektów różnych typów (błędy takie powinny być wychwytywane przez kompilator). Instrukcje mają dość standardową postać — istnieją procedury i funkcje. Operatorom można nadawać różne znaczenie w różnych częściach programu.

Jedną z najbardziej nowatorskich cech ADY jest pojęcie „pakietu" (ang. package), które jest rozszerzeniem dobrze znanego pojęcia procedury. Pakiety są wykonywalne, ale szczegóły dotyczące danych i procesów przez nie wykorzystywanych są ukryte przed użytkownikiem, którego jedynym zadaniem jest wywołanie pakietu. Ponadto pakiety służą do opisu wspólnych danych i w ten sposób spełniają rolę podobną do COMPOOL w JOVIALU lub Data Division w COBOLU. Pakiety można wykorzystywać także jako sposób uogólnienia danych.

Podprogramy i pakiety mogą być wzorcami części programu, a ich typy mogą być sparametryzowane. Przykładem takiej uogólnionej konstrukcji może być fragment programu, dla którego — jako chwilowe wartości danych wejściowych — dopuszczalne są skalary, wektory lub macierze. Dzięki wbudowanym mechanizmom (np. typ obiektów CLOCK — zegar) możliwe jest wykonywanie procesów współbieżnych oraz działanie w czasie rzeczywistym. Istnieje silny system przerwań i obsługi sytuacji wyjątkowych. Dozwolona jest niezależna kompilacja procedur. Wejście-wyjście obsługiwane jest przez bardzo prymitywne operacje, a bardziej skomplikowane czynności mają być wykonywane poprzez tworzenie pakietów. Wydajność programu można zwiększyć przez użycie deklaracji określających wewnętrzną reprezentację obiektu, co jednak zmniejsza przenośność programu.

Prognozy

Zdaniem autorki, ADA jest językiem mocniejszym i bardziej złożonym niż wymagał tego dokument IRONMAN. Jest to niewątpliwie szczytowe osiągnięcie końca lat siedemdziesiątych w dziedzinie projektowania języków. Przez wiele następnych lat studenci, projektanci języków, informatycy innych specjalności i, oczywiście, użytkownicy będą starali się zrozumieć wszystkie niuanse ADY. Całą złożoność tego języka i konsekwencje jego stosowania będzie można ocenić jednak dopiero, gdy poznamy działanie jego pierwszych kompilatorów (1982—1983).

Autorka uważa, że projektowanie ADY przeprowadzono w najlepszy z możliwych sposobów: opinie były zbierane podczas wszystkich etapów pracy, a decyzje dotyczące jego ostatecznej postaci zostały podjęte (po wnikliwych konsultacjach) przez bardzo małą grupę osób.

Opracowały:
TERESA WOJCIEKIAN
HALINA CIECHOMSKA
na podstawie referatu
Jean E. Sammet z konferencji
SEAS AM'82

Stały kontakt z INFORMATYKĄ gwarantuje tylko prenumerata

Zasady jej zamawiania — na stronie 29

COMPCONTROL'83

Podobnie jak poprzednie, VI międzynarodowa konferencja COMPCONTROL 83 w Bratysławie obejmowała szeroką problematykę zastosowań komputerów w przemyśle maszynowym. Zarówno obrady plenarne, jak i prezentacja rozwiązań szczegółowych metodą POSTER — odbywały się w trzech sekcjach:

- Zastosowanie komputerów w zarządzaniu przedsiębiorstwem
- Zastosowanie komputerów w konstrukcyjno-technologicznym przygotowaniu produkcji
- Rozłożone systemy sterowania.

Konferencja zgromadziła informatyków z krajów RWPG (w tym ponad 50 osób z Polski) oraz z krajów wysokorozwiniętych (USA, Japonii, RFN, Francji i in.).

Porównując z poprzednimi konferencjami (1977 — w Polsce, 1979 — na Węgrzech, 1981 — w Bułgarii), konferencja w Bratysławie była mniejsza, co być może przejawia się pewnym kryzysem tego typu zgromadzeń. Wpływ na ten stan rzeczy ma stosunkowo powolny postęp w zastosowaniach komputerów w zarządzaniu, a także brak poważniejszego wzajemnego zainteresowania osiągnięciami krajów RWPG, co przejawiło się niewielkim „natężeniem" rozmów kuluarowych oraz tworzeniem się nielicznych grup dyskusyjnych podczas szczegółowych prezentacji.

Konferencję można jednak nazwać konferencją *przewartościowania* pewnych pojęć związanych ze strategią i taktyką wdrażania informatyki.

*

Historia zastosowań komputerów oraz poglądów na taktykę ich stosowania w przedsiębiorstwach liczy już sobie ok. 30 lat [4, 9]. W okresie tym można wyróżnić trzy fazy:

- Integracja rozumiana jako *centralizacja* przetwarzania danych i sterowania produkcją. Podstawowe funkcje „zautomatyzowanego" zakładu jakimi obciążono centralny zestaw komputerowy nie były, oczywiście obsługiwane przez żadne terminale, co nastąpiło dopiero w latach sześćdziesiątych. Koncepcja ta nie mogła się sprawdzić, choćby dlatego, że:
  — potencjał obliczeniowy oraz niezawodność działania komputerów I i II generacji były znacznie „przeszacowane"
  — zapotrzebowanie na środki techniczne informatyki oraz na ich oprogramowanie były „niedoszacowane" (brak było odpowiednich systemów operacyjnych, języków programowania wyższego poziomu itp.).,
- W połowie lat sześćdziesiątych (w krajach wysokouprzemysłowionych) integracja zaczęła być rozumiana jako *centralna baza danych* niezbędnych do zarządzania i sterowania fabryką. Natomiast dla specyficznych zastosowań (np. obliczenia projektowo-konstrukcyjne, obrabiarki sterowane numerycznie, zbieranie danych źródłowych) przeznaczone były wyspecjalizowane komputery, które przekazywały dane do „banku centralnego" i stamtąd je również otrzymywały. Koncepcja ta wydawała się atrakcyjna i ekonomicznie uzasadniona: poszczególne podsystemy mogły być projektowane i wdrażane samodzielnie i dopasowywane do siebie — mając wspólny interfejs.

Mimo częściowego powodzenia tej koncepcji, okazało się, że integracja jest bardzo trudnym procesem z dwóch zasadniczych powodów:
— standaryzacja i unifikacja techniczna i organizacyjna przebiegały wolniej niż się spodziewano
— żaden system zarządzania bazą danych (SZBD) nie był w stanie obsłużyć efektywnie tak olbrzymiej masy danych; w praktyce przedsiębiorstw nie można skutecznie administrować tak szeroko pojętymi bazami danych, obejmującymi wszystkie sfery zarządzania i sterowania przedsiębiorstwem.

- W połowie lat siedemdziesiątych, wraz z pojawieniem się minikomputerów i mikroprocesorów, które umożliwiły zastosowanie *przetwarzania rozłożonego*, rozpoczęła się trzecia faza. Dostępność tanich i wyspecjalizowanych środków technicznych umożliwiły przeniesienie „inteligencji" komputerowej do poszczególnych dziedzin działalności lub funkcji przedsiębiorstwa. Początkowo były to układy typowo *hierarchiczne* (naśladujące hierarchiczną, wieloszczeblową strukturę systemu zarządzania lub sterowania). Praktyka zmusiła jednak do wypracowania bardziej skomplikowanych układów zależności.

Symptomatyczne jest, że w krajach przodujących w rozwoju przemysłu i zastosowań informatyki (w tym robotów) używa się skromniejszej terminologii niż w krajach będących na początku fazy powszechnej komputeryzacji [1]. Warto przytoczyć określenia anglosaskie [1, 9]:
CAD — Computer-Aided Design — komputerowo wspomagane projektowanie (wyrobów i części)

CAM — Computer-Aided Manufacturing — komputerowo wspomagane wytwarzanie, a więc sterowanie procesami technologicznymi
CMS — Computerised Manufacturing System — system skomputeryzowanej produkcji

[1] Symbole ASU (ZSRR, Bułgaria) lub ASŘ (CSRS) — to Zautomatyzowane Systemy Zarządzania (ZSZ)

CIM — Computer Integrated Manufacturing — komputerowo zintegrowana produkcja

CAPM — Computer-Aided Production Management — komputerowo wspomagane zarządzanie produkcją
ACS — Adaptive Control Systems for Manufacturing — adaptacyjne systemy sterowania produkcją.

Konkretna treść tych akronimów jest różna w poszczególnych branżach przemysłowych, np. samochodowej czy lotniczej. Większość przodujących fabryk na Zachodzie jest obecnie na etapie wdrażania systemów typu CAD i CAM wykraczających poza tradycyjne zastosowania ekonomiczno-statystyczne, określane jako DP (Data Processing).

Szeroko pojęta automatyzacja sterowania procesami technologicznymi traktowana jest jako kompleksowa *innowacja techniczna* i organizacyjna. Wymaga zaprojektowania nie tylko nowego systemu obiegu informacji (dokumentów) i metod ich przetwarzania (jak to się często u nas rozumie), ale całkiem nowych linii technologicznych (produkcji, transportu i składowania). Nowa technologia wymaga zastosowania innych (wspomaganych komputerowo) metod konstrukcyjnego i technologiczno-warsztatowego przygotowania produkcji. Tak przygotowana i oprzyrządowana produkcja jest, oczywiście, łatwa do zorganizowania i sterowania, a następnie — do zarządzania.

Wynika stąd prosty wniosek, że informatykę należy projektować kompleksowo już w momencie projektowania budowy lub modernizacji fabryki (wydziału), a nie przyklejać do istniejącej technologii i organizacji. Podstawowym motorem napędowym postępu jest nie pogoń za abstrakcyjnie rozumianą nowoczesnością, ale twarda walka konkurencyjna na rynku.

Wracając do problematyki terminologicznej — należy stwierdzić, że nie ma wyraźnej granicy między systemami CAD a CAM, czyli między konstrukcją i produkcją, mimo istnienia wydzielonych organizacyjnie biur projektowo-konstrukcyjnych. Wielu fachowców uważa (ponownie!), że jedna baza danych konstrukcyjno-technologicznych jest wystarczająca na potrzeby sterowania produkcją (jeżeli jest dobrze zaprojektowana i zrealizowana). Baza taka (tworzona etapowo) musi obsłużyć następujące czynności:

— projektowanie wyrobów i części (z możliwością ich obracania w trzech wymiarach)
— analizę „technologiczności" przyjętych rozwiązań konstrukcyjnych
— tworzenie dokumentacji graficznej
— planowanie produkcji, potrzeb materiałowych itp.
— programowanie obróbki części
— projektowanie narzędzi i oprzyrządowania oraz modeli
— kontrolę jakości produkcji.

Stąd już mały krok do komputerowego zarządzania. Wymaga to jednak podejścia systemowego: fabrykę trze-

ba widzieć jako całość (nie jako konglomerat poszczególnych elementów), a produkcję — jako proces uporządkowany (nie jako zestaw stosunkowo niezależnych działań).

Zastanawiające jest, że nawet w USA uważa się, że wytworzenie systemów CIM lub ACS jest poza zasięgiem możliwości jednej fabryki lub branży przemysłowej .

*

Mimo pewnego wyprzedzenia w stosunku do innych krajów socjalistycznych, autorzy czechosłowaccy [6] określają osiągnięty poziom informatyki w CSRS jako wyjściowy do powszechnej automatyzacji przetwarzania danych zarówno na potrzeby zarządzania, jak i sterowania produkcją. Na lata osiemdziesiąte założony jest etap rozwojowy, a etap zintegrowany (docelowy) rozpocząć się może dopiero w latach dziewięćdziesiątych.

Osiągnięty poziom zastosowań oceniony jest bardzo krytycznie, a uwagi niektórych autorów [7] mają charakter uniwersalny:

— automatyzację wiązano dotąd z przetwarzaniem danych ewidencyjno-statystycznych w konsekwencji czego skomputeryzowany został tylko proces podejmowania prostych decyzji gospodarczych

— istnieje wyraźna przewaga wsadowego przetwarzania danych w długich cyklach (przeważnie miesięcznych) — nawet cykl tygodniowy jest wyjątkiem; podobną sytuację przewiduje się w etapie rozwojowym ze względu na brak odpowiedniego sprzętu komputerowego

— „bazy danych" zorganizowane są w formie oddzielnych (częściowych) zbiorów danych, zorganizowanych z reguły sekwencyjnie

— utrudnione współdziałanie pracowników zarządu przedsiębiorstwa z komputerem realizowane nadal metodami tradycyjnymi (ręczne uzupełnianie dokumentów, z których dane przenoszone są przez operatorki z reguły na karty dziurkowane, na wyjściu tabulogramy)

— brak kompleksowości we wdrażonych podsystemach, eksploatowane systemy przetwarzania danych mają charakter wybitnie statyczny

— istniejące metody i techniki organizacji i zarządzania z reguły nie uległy zmianie w wyniku stosowania komputerów (w dziedzinie sterowania produkcją przewagę ma podejście empiryczno-dyspozytorskie — które staje się hamulcem intensywnego rozwoju produkcji).

W CSRS planowane jest przyspieszenie prac w dziedzinie automatyzacji procesów technologicznych w oparciu o mini- i mikrokomputery własnej produkcji (SM 4-20, SM 52/11 oraz SM 53/10-20). Niestety, również pozostałe, niezbędne do automatyzacji urządzenia, takie jak transport technologiczny, urządzenia pomiarowe i regulacyjne, mają być produkowane w CSRS, co niestety źle świadczy o procesach integracyjnych w RWPG.

Aktualnie pracuje w Czechosłowacji kilka eksperymentalnych wydziałów produkcyjnych stosujących niektóre elementy sterowania produkcją prostych elementów, np. konstrukcji skrzyniowych. Osiągnięty tam poziom automatyzacji niestety nie jest zbyt wysoki. Uzyskane doświadczenia wskazują, że automatyzacja sterowania procesami jest problemem bardzo złożonym [7]:

— stabilność systemów sterowania jest do pewnego stopnia iluzją

— metody sterowania podlegają podobnym regułom procesu innowacyjnego jakie występują w konstruowaniu maszyn i technologii ich produkcji

— nieodłącznym atrybutem każdego systemu sterowania jest jego adaptacja (przystosowanie do sytuacji nieoczekiwanych)

— komputery przeznaczone do sterowania produkcją muszą być niezawodne oraz mieć pamięć operacyjną i pomocniczą o znacznej pojemności

— kłopotliwy jest podział pracy (specjalizacja), a także koordynacja w procesie przygotowania systemu sterowania produkcją.

Wybrane referaty VI konferencji COMPCONTROL 83:

1. Malec I.: Jake technické prostředky vytváří čsl. elektrotechnicky průmysl pro to, aby jeho počitače mohly sloužit pro řizení procesu?

2. Guedj R. A.: Human-Machine Systems: Will We Provide A Congenial Environment For The Engineering?

3. Hantúch J.: GPSR — prostriedek pre automatickú syntézu programového systému riadenia v reálnom čase pre operačny systém DOSRV2

4. Hatvany J.: Integrated Design, Manufacturing And Testing Systems

5. Havlík J.: Programové wybavení pro dialogový režim práce na minipooi-tači ADT

6. Kozary Z.: Smery budování automatizovaných systému rizeni v československém strojirenstvi

7. Malec I.: Jaké technické prostředky vytváří čsl. elektrotechnický průmysl pro to, aby jeho počitače mohly sloužit pro řizení procesů?

8. Novák J.: Algoritmus rozvrhování složité strojírenské výroby

9. Olling G.: New Trends In The Computer Control Of Manufacturing Operations

10. Ružička F.: Uživatelské přizbusobování APV užité pro řizení VDF v podmínkách technického a programového wybavení SMEP

11. Warnecke H. J.; Walther J., Bässler R.: The Automation Of Assembly A Major Break-Through

12. Werler K. H.: Einige Probleme der Weiterentwicklung des Systems der Rationalisierung der Produktion (TEVO) und die Relation zum Prozess der Fertigung.

Istotne jest również doświadczenie specjalistów czechosłowackich związane ze standaryzacją rozwiązań projektowo-programowych w zakresie systemów sterowania produkcją. Otóż, mimo narzuconej jednolitej metodyki, każdy system jest niestety robiony „na miarę"; nie ma bowiem dwóch identycznie zorganizowanych fabryk. Istotne znaczenie ma więc podział zadań w przygotowaniu typowego (wzorcowego) oprogramowania, umożliwiającego powielanie i adaptację. Poszczególne czechosłowackie ośrodki badawczo-rozwojowe oprogramowują obecnie różne typy mikroprocesorów lub różne funkcje sterowania, np. obsługę magazynu wysokiego składowania.

O ile doświadczenia i zamierzenia CSRS w zakresie przygotowywania i wdrażania systemów sterowania produkcją z zastosowaniem mikro- i minikomputerów mają charakter uniwersalny, o tyle referaty dotyczące automatyzacji resortowych lub zjednoczeniowych systemów zarządzania raziły swoim utopizmem (przynajmniej w tym zakresie jesteśmy bogatsi w doświadczenia i wiemy, że to nie wyjdzie, że nie ma praktycznego sensu), ponieważ tego rodzaju próby komputeryzacji nie są podbudowane żadnymi pracami teoretycznymi lub praktycznymi w zakresie systemu zarządzania gospodarki narodowej.

*

Poza referatami wygłoszonymi na obradach plenarnych lub w sekcjach odbyły się liczne prezentacje szczegółowych rozwiązań metodą POSTER (plansze wywieszone w specjalnych boksach). Specjaliści zainteresowani konkretną problematyką mogli przedyskutować z autorami tych ekspozycji konkretne osiągnięcia aplikacyjne. Ze względu na znaczną liczbę tych prezentacji (ok. 100) — trudno je w całości omówić. Można jedynie ocenić, że rozwiązania specyficzne, np. zastosowanie małego komputera produkcji węgierskiej R 10M do krótkookresowego operatywnego planowania produkcji — miały jedynie lokalne znaczenie. Podobna uwaga odnosi się do zastosowania unikalnych w skali RWPG, importowanych z Zachodu systemów komputerowych np. SIEMENS 3300, R 800 itd.

Bardziej obiecujące są doświadczenia zdobyte na sprzęcie produkowanym w krajach RWPG a więc na wspomnianych minikomputerach SM-4 lub SM 52/11 [2]). Niestety, bez większego echa przeszła prezentacja generatora programów do sterowania produkcją [3, 10]. Rzecz charakterystyczna — referaty i prezentacje autorów z CSRS, Węgier i Polski miały charakter praktyczny natomiast re-

[2]) W najbliższych latach planuje się uruchomienie produkcji zestawu SM 52/11 PLUS z pamięcią operacyjną o bardzo dużej pojemności (do 4 MB!) oraz dyskami 100 MB; poza tym już obecnie CSRS produkuje minikomputer ADT 4500 z pamięcią operacyjną o pojemności do 2 MB 5.

feraty autorów z ZSRR dotyczyły głównie problematyki teoretycznej. Podobny charakter miały niektóre referaty autorów z Bułgarii i NRD.

*

Problematyka sterowania produkcją montażową została interesująco opisana w referacie autorów z RFN [11]. Pożyteczne są ich wnioski:

- istnieje wiele zastrzeżeń do automatyzacji produkcji montażowej z zastosowaniem robotów przemysłowych, a mianowicie:

— duża awaryjność robotów i brak fachowców do ich regulacji oraz naprawy

— brak miejsca w istniejących halach produkcyjnych na instalowanie robotów

— nie jest dopracowana technologia sensorów

— brak jest doświadczeń w wykorzystaniu robotów przemysłowych

— duży koszt kompletnego systemu automatyzacji produkcji (szacuje się, że koszt samych robotów sięga 30%, a resztę stanowi koszt ich oprzyrządowania, czyli urządzeń peryferyjnych)

— mała podaż kompletnych systemów automatyzacji.

W najbliższej przyszłości pojawi się także wiele problemów w upowszechnianiu się automatyzacji produkcji:

— należy rozwiązać problem przygotowania i podawania części do montażu

— aktualne systemy nie są dostosowane do montażu finalnego (problem współpracy z operacjami ręcznymi)

— pojęcie „montażu" oznacza nie tylko czyste łączenie części, ale również wiele operacji pomocniczych, zwłaszcza kontrolnych, wykonywanych przez człowieka

- największym problemem do rozwiązania jest jednak niedostosowanie rozwiązań konstrukcyjnych wyrobów do ich automatycznego montażu (autorzy podają szereg ogólnych wskazówek jak problem ten złagodzić).

*

Ze względu na szczupłe ramy niniejszej relacji nie zostaną omówione inne referaty, w tym — autorów polskich, mimo że ich obecność na konferencji była znacząca.

JERZY SUKIENNIK
Gdynia

Dotykać komputer

Postęp w dziedzinie komputerów osobistych mierzy się w tej chwili już nie latami, lecz miesiącami. Niedawno firma APPLE oficjalnie przedstawiła swój komputer LISA, który wyróżnia się możliwością wprowadzania programów nie tylko przez klawiaturę, ale też dzięki ruchom specjalnego manipulatora, który jest sprzężony z kursorem na ekranie. W połączeniu z inteligentnym systemem operacyjnym manipulator (nazywany „myszką" od swego kształtu) pozwala na bardzo szybkie operowanie standardowymi procedurami typu wprowadzania danych, manipulowania tabulogramami, a nawet programowania blokowego. Niestety, LISA kosztuje 7 tys. dol., a niebagatelną część tej sumy stanowi koszt inteligentnego oprogramowania komputera.

Tymczasem we wrześniu ub.r. firma HEWLETT-PACKARD ogłosiła wypuszczenie nowego modelu komputera osobistego: HP 150. Nowością w tym przypadku jest zastosowanie ekranu monitora, który służy także do wyboru komend systemowych poprzez: dotykanie.

HP Touch Screen w rzeczywistości wcale nie wymaga dotknięcia. Działa on bowiem na zasadzie przerywania strumienia światła podczerwonego. Ekran komputera został obramowany dwoma zestawami: diody świecące — fotodiody. Każdorazowo przecięcie tego rastra świetlnego powoduje określenie położenia obiektu zakłócającego. Dokładność tej metody nie jest zbyt duża (rozdzielczość zależy od średnicy obiektu zakłócającego oraz od liczby linii rastra — zastosowano 24 na 40 linii, przy czym kolumny są dwuliniowe), lecz w zupełności wystarcza do rozróżnienia poszczególnych pól rozkazów ekranowych.

Komputer HP 150 został zbudowany w oparciu o mikroprocesor INTEL 8088 pracujący przy 8 MHz (zamiast standardowych 5 MHz — co pozwala na szybsze przetwarzanie danych) i ma pamięć 256 KB z możliwością rozszerzenia do 640 KB. W komputer wmontowano dwa napędy dysków elastycznych o średnicy $3\frac{1}{2}$ cala (tzw. mikrodyskietki), dwa sprzęgi RS-232 oraz jeden w systemie HP-IB. Monitor ma przekątną 9 cali i pozwala na wyświetlanie 24 linii po 80 znaków plus trzy dodatkowe linie na status systemu. Oczywiście kolor ekranu jest zielony — dla ułatwienia dłuższej pracy (nie męczy oczu).

Aby HP 150 mógł konkurować z LISĄ, został wyposażony w bardzo bogate oprogramowanie systemowe. System operacyjny MS-DOS 2.0 umożliwia użytkowanie programów napisanych przez niezależnych programistów. Właściwy system operacyjny komputera (nazwany Personal Applications Manager) pozwala na całkowite oddzielenie użytkownika od wewnętrznych komend systemowych — użytkownik może, dotykając ekran, wybierać różne programy, kopiować pliki danych oraz dyskietki, a także tworzyć swoje własne bazy danych. Wreszcie komputer został wyposażony w specjalny program, który uczy użytkownika jak posługiwać się komputerem, a w szczególności — jak dotykać ekranu.

Oczywiście HP 150 został wyposażony w firmowe oprogramowanie pozwalające na bardzo różnorodne zastosowania komputera. Tak więc użytkownik ma do dyspozycji program WordStar, ułatwiający pisanie tekstów, oraz dziś już klasyczny VisiCale, ułatwiający formowanie wszelkich tabel. Ponadto HP 150 został wyposażony w pakiet graficzny (też z możliwością wyboru elementów obrazu przez dotykanie ekranu).

Standardowym językiem jest MICROSOFT BASIC, ale firma HEWLETT-PACKARD przewiduje wypuszczenie dysków z PASCALEM. Cały system (jednostka centralna z procesorem 8088, monitorem, dwoma napędami dyskietek oraz z opłaconym przewozem do użytkownika) kosztuje 4000 dol.

Czyżby firmie APPLE i jej LISIE zagroziła konkurencja?! W każdym razie można się spodziewać, że niebawem kolejne firmy wypuszczą swoje odmiany komputerów „dla ludzi". Obyśmy także w Polsce mogli się ich doczekać.

Oprac. J.T.
na podstawie materiałów firmy HEWLETT-PACKARD

DO CZYTELNIKÓW

Poszukujemy unikalnych materiałów informacyjnych związanych ze światową informatyką. Chętnie wypożyczymy na krótko pojedyncze numery czasopism, kopie artykułów, reklamówki, katalogi itp. Pomóżcie nam dotrzeć do najnowszych źródeł, a zrewanżujemy się atrakcyjniejszym pismem. (Red.)

MEDINFO'83

Czwarty światowy Kongres Informatyki Medycznej MEDINFO'83 odbył się w dniach 22—26 sierpnia 1983 w Amsterdamie. To najwyższe forum prezentacji osiągnięć informatyki medycznej w zakresie sprzętu i oprogramowania było poprzedzone trzema kongresami (w Szwecji, Kanadzie i Japonii), które odbywały się co trzy lata. Obecnie MEDINFO'83 znalazło dogodne dla siebie miejsce w nowoczesnym centrum kongresowym RAI, usytuowanym w samym śródmieściu Amsterdamu. W Kongresie wzięło udział ok. 1500 osób, w tym ok. 930 uczestników akredytowanych. Liczebność delegacji wahała się od ponad 100 (Holandia, USA) do kilku osób (kraje socjalistyczne) i nie była wyznacznikiem poziomu zainteresowania tą dziedziną, a możliwością sfinansowania wyjazdu. Dla porównania, z czteromilionowej Finlandii przybyło 28 uczestników.

Obrady Kongresu zorganizowano w następujących czterech płaszczyznach: sesje naukowe, spotkania dyskusyjne (ang. workshops), seminaria oraz wycieczki.

Sesje naukowe odbywały się równolegle w czterech salach i objęły w ciągu pięciu dni 24 posiedzenia. Tematyka poszczególnych posiedzeń dotyczyła m.in. takich zagadnień, jak: informacyjne systemy otwartej opieki zdrowotnej, szpitalnictwa i zaopatrzenia w leki, terenowe i ogólnokrajowe systemy służby zdrowia, komputerowa analiza obrazów, komputerowe wspomaganie decyzji klinicznych, badań naukowych i szkolenia zawodowego, przetwarzanie biosygnałów, monitorowanie pacjentów, automatyzacja badań laboratoryjnych, poufność i zabezpieczanie danych medycznych, statystyka medyczna. Wystąpienia prelegentów odbywały się w formie wygłoszenia limitowanych czasowo referatów lub komunikatów. Wygłoszono łącznie ponad 300 referatów oraz ponad 60 komunikatów. Polskę reprezentowały: referat („Relacyjna baza danych medycznych — doświadczenia i zamierzenia na przyszłość" — autor P. Jasiński z Politechniki Poznańskiej) oraz komunikat („Projekt komputerowego nadzoru cyklu ciążowego" — autorzy: K. Gwóźdź i L. Wolański z ZETO Wrocław).

Spotkania dyskusyjne, wprowadzone do programu Kongresu po raz pierwszy w 1980 r. w Tokio, odbyły się w drugim i czwartym dniu obrad, po zakończeniu posiedzeń sesji naukowych. Spotkania prowadzone były podobnie jak sesje równolegle w czterech salach i objęły osiem najważniejszych, zdaniem organizatorów, tematów informatyki medycznej:

- otwarta opieka zdrowotna
- kształcenie kadr
- formułowanie hipotez medycznych na podstawie pośrednich obserwacji klinicznych
- komputerowe przetwarzanie obrazów
- nowe technologie w systemach informacyjnych
- ogólnokrajowe planowanie działalności służby zdrowia
- sztuczna inteligencja w medycynie
- metody projektowania i wdrażania systemów oddziałowych.

Seminaria, za uczestnictwo w których obowiązywały dodatkowe odrębne opłaty miały nieco inny cel, a mianowicie zapoznanie określonych grup użytkowników z postępem w interesujących ich dziedzinach zastosowań. Seminaria stanowiły całkowicie nowy element w programie Kongresu, a ich treść prezentowali czołowi specjaliści z różnych krajów. Seminaria, rozłożone na 1—3 dni odbywały się również równolegle do podstawowego nurtu Kongresu. Organizatorzy oferowali do wyboru sześć seminariów na następujące tematy:

- planowanie i zarządzanie
- systemy informacyjne lecznictwa zamkniętego
- automatyzacja zaopatrzenia w leki
- komputer w gabinecie lekarza
- komputeryzacja laboratoriów klinicznych
- systemy opieki pielęgniarskiej.

Podobny charakter miały tzw. sesje specjalne (ang. special sessions) wprowadzone po raz pierwszy do programu Kongresu z inicjatywy niektórych krajów lub organizacji. Ich treść nie była weryfikowana przez Komitet Programowy, a tematyka dotyczyła ogólnych aspektów rozwoju medycznych baz danych oraz rozwoju systemów informatycznych służby zdrowia w Europejskiej Wspólnocie Gospodarczej, a także koncepcji i realizacji takich systemów we Włoszech i Wielkiej Brytanii.

Wizytacje medycznych systemów informatycznych umożliwiały zapoznanie się ze zrealizowanymi w Holandii rozwiązaniami. Podobnie jak w przypadku seminariów, uczestnictwo wymagało wniesienia dodatkowych, tym razem stosunkowo niewielkich opłat na koszty przejazdu autobusem (10—25 florenów). Istniała możliwość zapoznania się z działalnością i osiągnięciami następujących placówek:

- kliniki uniwersyteckiej w Leiden
- ośrodka przeciwzakrzepowego w Amsterdamie
- wydziału systemów medycznych firmy PHILIPS w Eindhoven
- laboratorium histo- i cytochemicznego oraz Holenderskiego Instytutu Prewencyjnej Opieki Zdrowotnej w Leiden
- wydziału informatyki medycznej Wolnego Uniwersytetu w Amsterdamie
- centrum medycyny tropikalnej Wydziału Medycyny oraz Kliniki uniwersyteckiej w Rotterdamie.

GIEŁDA INFORMACJI

Publikujemy bezpłatnie:

- **krótkie prezentacje nowych, nie wykorzystanych dostatecznie systemów**
- **doniesienia o innowacjach programowych**
- **ogłoszenia dotyczące nowych rozwiązań technicznych sprzętu informatycznego**
- **indywidualne oferty informatyków poszukujących pracy oraz informacje o wolnych etatach**
- **ogłoszenia w sprawie szkoleń informatycznych**
- **inne informacje ułatwiające doraźną organizację informacji w Polsce.**

Emulator ODRY 1305 na R-32

ZETO Gdańsk oferuje emulator będący narzędziem umożliwiającym bezpośrednie przetwarzanie programów ODRA na maszynie R-32. Emulator powoduje, że komputer R-32 staje się całkowicie funkcjonalnie zgodny z maszyną ODRA 1305 pracującą pod egzekutorem EGRM wydanie 44, jak i z dowolną maszyną Jednolitego Systemu pracującą pod symulatorem E6RM-S, oferowanym również przez ZETO Gdańsk.

Emulator jest czterokrotnie szybszy od symulatora, a prędkość przetwarzania jest większa od uzyskiwanej na maszynie ODRA 1304. Przyspieszenie to zostało osiągnięte dzięki dodaniu dodatkowych programów do pamięci stałej R-32.

Emulator współpracuje tylko z maszynami R-32 od numeru procesora 085. Cena zainstalowania wynosi 850 tys. zł, zaś dla posiadaczy symulatora E6RM-S — 320 tys. zł.

Zamówienia można kierować pod adresem: ZETO Gdańsk, ul. Żwirki i Wigury 15, 81-869 Gdynia; tel. 21-66-46; teleks 654258.

LABADA

Instytut Informatyki Akademii Ekonomicznej we Wrocławiu opracował interaktywne laboratorium wspomaganego komputerem projektowania baz danych LABADA — zestaw czternastu programów wspomagających pracę analityka, projektanta lub programisty systemów informatycznych. Programy wchodzące w skład laboratorium mogą pracować na komputerach Jednolitego Systemu z min. 512 K bajtów pamięci operacyjnej oraz systemem TSO.

Programy mogą realizować: analizę statystyczną istniejących zbiorów danych, modelowanie procesów kodowania i kompresji danych, modelowanie procesów randomizacji, obliczanie optymalnego współczynnika blokowania, obliczanie obszarów zbiorów w pamięci dyskowej i generowanie parametrów zdania DD języka opisu zadań (dla systemu OS/JS), modelowanie organizacji zbiorów danych (sekwencyjnej, in-

→

GIEŁDA INFORMACJI

deksowo-sekwencyjnej, indeksowanej, wielolistowej i bezpośredniej), obliczanie parametrów skorowidzów typu B-drzewo oraz modelowanie zbiorów przyrostowych. Ponadto realizowane są prace rozwojowe nad wielowariantowym programem hierarchicznych baz danych oraz pakietem wspomagającym konstruowanie struktury logicznej takiej bazy.

Dalszych informacji udziela: **dr Janusz Wojdyła: Instytut Informatyki, Akademia Ekonomiczna, ul. Komandorska 118/120, 53-345 Wrocław; tel. 68-11-55, w. 378, 377.**

System Zarządzania Produkcją

ZETO Wrocław oferuje System Zarządzania Produkcją na mikrokomputer SM-4 „PMS", pomyślany jako zintegrowany system „on-line", stanowiący rozwinięcie istniejącego systemu tego typu na minikomputerze DATA-POINT 5500/6600. System przeznaczony jest dla przedsiębiorstw przemysłowych średniej wielkości o produkcji seryjnej.

Główne cechy systemu:

— zintegrowana baza danych eliminująca zjawisko dublowania się informacji

— łatwość obsługi przez personel nie posiadający przygotowania informatycznego — dzięki zastosowaniu listy funkcji kierujących użytkownika do właściwego programu oraz serii dyrektyw specyfikujących dokładne czynności, jakie ma wykonać

— pełna kontrola i monitowanie przepływu materiałów przez zakład

— zastosowanie sprawdzonych technik zarządzania produkcją.

● Zastosowanie systemu usprawnia proces planowania produkcją poprzez:

— szybkie wprowadzenie zmian do technologii i struktury wyrobów

— usprawnienie gospodarki materiałowej i uzyskanie przez to większej dyspozycyjności materiałów z równoczesnym zmniejszeniem zapasów i lepszym ich wykorzystaniem

— lepszą obsługę zamówień odbiorców (na wyroby i części zamienne)

— szybkie wprowadzanie zmian w kosztach normatywnych wyrobów oraz możliwość śledzenia kosztów rzeczywistych

— lepsze wykorzystanie parku maszynowego i siły roboczej przedsiębiorstwa.

● Minimalny zestaw technicznych środków eksploatacji systemu

— przewiduje się standardowy zestaw minikomputera SM-4 w wersji terminalowej z dyskami magnetycznymi o pojemności 30—60 MB.

Zgłoszenia przyjmuje Kierownik Działu Marketingu, mgr Tadeusz Czerniewski; Zakład Elektronicznej Techniki Obliczeniowej, ul. Ofiar Oświęcimskich 7/13, 50-069 Wrocław; tel. centrali 44-54-31 do 37; Teleks 0712533 ZETO PL.

Zgodnie z tradycją Kongresowi towarzyszyły pokazy i wystawa, w ramach której liczne firmy komputerowe demonstrowały nie tyle sprzęt (głównie mikrokomputery), ile oprogramowanie zastosowań medycznych. Wśród wystawców można było dostrzec takie firmy, jak: OLIVETTI, BURROUGHS, PHILIPS, SIEMENS, HONEYWELL, HEWLETT-PACKARD.

Organizatorzy przewidzieli również dla uczestników Kongresu szereg odpłatnych imprez towarzyszących, jak wycieczki po Amsterdamie i okolicach.

Uczestnicy Kongresu otrzymali starannie wydany pod względem poligraficznym 2 tomowy zbiór materiałów Kongresu o objętości około 2300 stron.

Polskich uczestników uderzyła szczególnie wielka podaż nowoczesnego zminiaturyzowanego sprzętu komputerowego o bardzo różnorodnej i bogatej konfiguracji, dostosowanej do specyfiki pracy lekarza, niskie ceny i dogodne warunki dostawy tego sprzętu, a także bujny rozwój oprogramowania z zakresu zastosowań medycznych. Odnosiło się przekonanie, że w świadomości uczestniczących lekarzy informatyka uzyskała już pełne prawo obywatelstwa oraz zrozumienie dla niezbędności powszechnych wdrożeń jako warunku sprawnej i skutecznej opieki zdrowotnej we współczesnym społeczeństwie.

Miejscem następnego Kongresu (MEDINFO'86) będzie Waszyngton (USA).

LESŁAW WOLAŃSKI
KAZIMIERZ GWÓŹDŹ
ZETO Wrocław

EUROMICRO

EUROMICRO (European Association for Microprocessing and Microprogramming) jest niezależnym, niedochodowym stowarzyszeniem kierowanym przez międzynarodową radę pochodzących z wyboru dyrektorów. Jego celem jest rozszerzanie wymiany informacji w dziedzinie mikroprogramowania z uwzględnieniem zarówno technicznych, jak i nietechnicznych aspektów. EUROMICRO wydaje miesięcznik „Microprocessing and Microprogramming — The Euromicro Journal" i organizuje doroczne Sympozjum Euromicro oraz wspiera i współorganizuje również inne podobne przedsięwzięcia.

Członkowie EUROMICRO automatycznie otrzymują miesięcznik oraz są informowani o wszystkich wydarzeniach. Przysługuje im zniżka w opłatach za uczestnictwo we wszystkich spotkaniach organizowanych lub współorganizowanych przez stowarzyszenie. Wszyscy członkowie są zachęcani do aktywnego, a nie tylko formalnego uczestnictwa w organizacji.

EUROMICRO powstało w 1973 roku. Początkowo — od października 1974 — wydawano kwartalnik „Euromicro Newsletter", który w 1978 stał się dwumiesięcznikiem „Euromicro Journal". Dzisiejszą postać i tytuł pismo stowarzyszenia otrzymało w 1981 roku.

Pierwsze Sympozjum odbyło się w 1975 roku w Nicei (Francja), a w kolejnych latach w Wenecji (Włochy), Amsterdamie (Holandia), Monachium (RFN), Göteborgu (Szwecja), Londynie (W. Brytania) i Paryżu (Francja). Przeciętnie spotykało się na nich ok. 500 osób — praktyków i teoretyków z przemysłu, jednostek rządowych i świata akademickiego. Tradycyjnie Sympozjum poprzedza jednodniowe seminarium prowadzone przez wybitnych specjalistów, a dotyczące szczególnie ważnych tematów (np. ADA, projektowanie układów VLSI, lokalne sieci danych). Główne obrady składają się z referatów, dyskusji, komunikatów i doniesień przemysłowych. Jednocześnie rozgrywa się finały zawodów EUROMOUSE (myszki-roboty szukają drogi w labiryncie) oraz odbywa się wystawa.

Kierująca pracami stowarzyszenia rada dyrektorów zbiera się co kwartał, za każdym razem w innym mieście europejskim. Dla zapewnienia równowagi w kierunkach prac, każdy kraj reprezentuje dwóch dyrektorów — jeden z przemysłu, drugi ze świata nauki. Chociaż EUROMICRO jest programowo niezależne od jakichkolwiek innych organizacji, to jednak utrzymuje ścisłą współpracę z podobnymi stowarzyszeniami, szczególnie z ACM i IEEE (USA), JEIDA (Japonia) oraz IEE (W. Brytania), a także z IFIP.

Członkiem EUROMICRO może zostać każdy — warunkiem jest opłacanie składki w wysokości 50 florenów rocznie.

EUROMICRO
T.H. Twente
P.O. Box 217
Dept. Informatics, Room A312
7500 AE Enschede
Holandia
telefon (953) 33-87-99
teleks 44200 thes

Oprac. MkS

W najbliższych numerach przedstawimy dalsze organizacje informatyczne: IFIP, BCS, AFCET, NCC, a także organizację krajów RWPG — Radę ds. Zastosowań Srodków Techniki Obliczeniowej. Czytelników, którzy chcieliby znaleźć na naszych łamach prezentację innych organizacji zajmujących się informatyką, prosimy o kontakt. (Red.)

„Przegląd mikroprocesorów 16-bitowych"

Ponieważ ani w numerze 4, ani 5/83 nie doczytałem się żadnego wyjaśnienia dotyczącego artykułu Jerzego Grabowskiego o rodzinie mikroprocesorów ZILOG Z8000 (INFORMATYKA nr 3/83 s. 2 — przyp. red.), pragnę poinformować Redakcję, że znalazły się w tym artykule — oraz w całym cyklu o mikroprocesorach 16-bitowych — istotne błędy merytoryczne.

Uważna lektura artykułu skłania mnie do stwierdzenia, że Autor nie posługiwał się dokumentacją firmy ZILOG, natomiast opisał uboższą wersję mikroprocesora Z8001!2, znaną pod nazwą AmZ8001/2, produkowaną przez firmę AMD. Przyczyn nie znam, ale prawdopodobnie Autor nie zna po prostu tych różnic, które sprawiają, że mikroprocesor Z8001 znakomicie radzi sobie z pracą w środowisku wieloprocesorowym, czego nie można powiedzieć o jego „odpowiedniku" firmy AMD.

Wydaje się, że w cyklu artykułów, który prawdopodobnie będzie miał duże konsekwencje w przyszłości — przy wyborze wyrobu mikroprocesora 16-bitowego do opracowywania (czytaj: kopiowania) w kraju, Autor powinien korzystać z publikacji oryginalnych, tym bardziej, że były one prezentowane na tegorocznych Międzynarodowych Targach Książki (Z8000 CPU TECHNICAL MANUAL lub Z8000 CPU USER'S REFERENCE MANUAL, Prentice-Hall, Inc, ISB 0-13-983890-2, C. 1982, 81, 80 by Zilog, Inc.). Dobrze byłoby nie popełniać takich błędów, jak wybór 8080, zamiast 8085 czy Z80 do produkcji w kraju.

Zaznaczam, że dotychczas wysoko ceniłem publikacje Jerzego Grabowskiego i sądzę, że w przyszłości nie spotkam podobnych nieścisłości w jego artykułach. A zatem do rzeczy.

Rysunek 1 pochodzi z opisu **AmZ** — powinno być na pozycji **b13 PSW** (dokładniej: **FCW**) nie **0**, lecz **EPA** (Extended Processor Architecture Mode). Brak objaśnienia pod **PC**, że chodzi o **PSAP** (rys. 1). Brak opisu **EPA instructions**. Zmiana jednego z przerwań zewnętrznych **(TRAPS)** z **Extended Instruction Trap** — symulowanie pracy procesorów wewnętrznych **EPU** przez program w Z8001, zamiast **Illegal Instr. Trap** w **AmZ.**

Co do wad przy pracy wieloprocesorowej Z8001 w porównaniu z 8086 — można dyskutować. Mikroprocesor Z8001 nie uwalnia szyn na czas pracy procesora zewnętrznego, ale pracuje z nim równolegle, komunikując się przez dwie linie **MI** i **MO** — jak **AmZ** — oraz pozwala procesorom komunikować się między sobą, pamięcią i procesorem centralnym Z8001, nie tracąc kontroli nad nimi, systemem oraz odświeżaniem pamięci dynamicznych RAM. Jeśli chce się utracić kontrolę nad szynami, to można posłużyć się linią MULTI-MICRO CONTROL OUT i podać z **EPU** sygnał **BUSRQ** lub użyć mikroprocesorów Z8003/4 — dodatkowe stany pracy: **Test Data Access** i **Set Data Access.** Poza **MREQ, MBIT, MSET, MRES,** Z8001 ma dodatkowe instrukcje sterujące przepływem informacji w systemie bez utraty nad nim kontroli: **Load Memory form EPU** i odwrotnie (normalnie **max. 4 Extended Processing Units), Leoad EPU Registers from CPU Registers** (1...16 rejestrów), to samo w drugą stronę, **Load FCW from EPU** i odwrotnie oraz **Internal EPU Operation** z programowanym czasem zwolnienia szyn. Tak więc procesor ten umożliwia rzeczywistą pracę wieloprocesorową, jeżeli za taką uznać pracę równoległą procesorów.

Warszawa, 26 listopada 1983

MICHAŁ CHOROSZUCHA

Od autora

Uwagi merytoryczne zawarte w liście Michała Choroszuchy są na ogół słuszne. Pragnę jednak wyjaśnić, że:

1. W artykule otwierającym Przegląd zaznaczyłem (być może nie dość jasno), że w opracowaniu akcent zostanie ustawiony na opis list rozkazów omawianych mikroprocesorów. W konsekwencji problemy związane ze współdziałaniem programu ze środowiskiem wielkoprocesorowym zostały potraktowane marginesowo.
2. Prawdą jest, że przygotowując artykuł nie znałem oryginalnych materiałów firmy ZILOG. Posługiwałem się głównie książką C. A. Titusa: 16-Bit Mikroprocesors, Howard W. Sams and Co., Indiana, 1981. Książka ta była dla mniej jedynym obszernym materiałem — dostępnym w czasie pisania artykułu, w roku 1982. Faktem jest też, że książka ta opisywała mutację mikroprocesora ZILOG 8000 produkowaną przez firmę AMD, co rzeczywiście uszło mojej uwadze.
3. Uważam, że autor listu przecenia wpływ mojego artykułu na podejmowanie decyzji przy uruchamianiu takich czy innych mikroprocesorów. Obserwując jednak skutki tych decyzji przypuszczam, że decydujący wpływ mają tu uwarunkowania (czytaj: trudności) technologiczne, a także uwarunkowania pozamerytoryczne.
4. Uwagi dotyczące pracy wieloprocesorowej mikroprocesorów Z8000 nie mają wewnętrznych sprzeczności, a więc nie pozostaje mi nic innego, jak uznać, że M. Choroszucha prawidłowo interpretuje opis firmowy. Pragnę jednak jeszcze raz powołać się na pierwszy punkt tego wyjaśnienia.

JERZY GRABOWSKI

Zasady prenumeraty

Zamówienia i przedpłaty na prenumeratę INFORMATYKI przyjmuje Zakład Kolportażu Wydawnictwa NOT SIGMA. Adres pocztowy: Wydawnictwo NOT SIGMA — Zakład Kolportażu, 00-950 Warszawa, skr. poczt. 1004. Konto bankowe: 1036-7490-139-11, III O/M NBP w Warszawie.

JEDNOSTKI GOSPODARKI USPOŁECZNIONEJ, INSTYTUCJE I ORGANIZACJE przesyłają zamówienia (w 1 egz.) zawierające: tytuł czasopisma, liczbę zamawianych egzemplarzy, okres prenumeraty i pełny adres zamawiającego z kodem pocztowym, oddział i nazwę banku z numerem konta bankowego zamawiającego oraz (ewentualnie) adres odbiorców, którzy na zlecenie i koszt zamawiającego mają egzemplarze otrzymywać.

Warunkiem realizacji zamówienia jest równoczesne dokonanie odpowiedniej wpłaty na ww. konto Wydawnictwa NOT SIGMA.

Za prenumeratę nie wystawiane są rachunki i nie potwierdzane salda. Prenumeratorzy zbiorowi proszeni są o podawanie na dowodach wpłat (przelewach) znaku kancelaryjnego zamówienia, którego dotyczy wpłata.

Dopisując na zamówieniu PRENUMERATA STAŁA, zamawiający (tylko prenumeratorzy zbiorowi!) nie będą musieli corocznie ponawiać zamówienia, a jedynie doknywać przedpłaty według aktualnie obowiązujących cen. Wydawnictwo przekazywać będzie co roku potwierdzenie kontynuacji prenumeraty.

PRENUMERATORZY INDYWIDUALNI dokonują wpłaty przekazem na ww. konto, pod powyższym adresem, podając na odwrocie odcinka dla adresata-posiadacza rachunku: tytuł czasopisma, liczbę zamawianych egzemplarzy oraz okres prenumeraty.

Do PRENUMERATY ULGOWEJ upoważnieni są członkowie stowarzyszeń naukowo-technicznych NOT, studenci, uczniowe szkół zawodowych. Warunkiem jej uzyskania jest poświadczenie blankietu dla nabywcy indywidualnego (na odcinku dla adresata) przez właściwe stowarzyszenie NOT, wyższą uczelnię lub szkołę zawodową.

Zamówienia i wpłaty przyjmowane są na okresy kwartalne, półroczne i roczne w terminach:

- do 15 listopada — na I kwartał, I półrocze i cały rok następny
- do 28 lutego — na II, III i IV kwartał
- do 31 maja — na IV kwartał i II półrocze
- do 31 sierpnia — na IV kwartał.

Uwaga: Przy podawaniu kodu pocztowego i numeru konta bankowego obowiązuje bardzo czytelne pismo. Prenumerata nie wymaga specjalnego przekazu z czerwonym paskiem; wystarczy zwykły przekaz bankowy.

Prenumerata normalna: kwartalna — 225 zł, półroczna — 450 zł, roczna — 900 zł. Prenumerata ulgowa: kwartalna — 150 zł, półroczna — 300 zł, roczna — 600 zł. Prenumerata ze zleceniem wysyłki za granicę jest dwukrotnie droższa.

Dodatkowych informacji o prenumeracie udziela: Zakład Kolportażu, tel. 40-00-21 w. 293, 299 oraz 40-35-89. Egzemplarze archiwalne można nabywać w Klubie Prasy i Informacji Technicznej w Warszawie, ul. Mazowiecka 12, tel. 27-43-65. Zamówienia na egzemplarze archiwalne należy kierować pod adresem Zakładu Kolportażu.

Kalendarz informatyczny

Nową formą wydawniczą — bo tak chyba najlepiej można określić **Mathematics and Computing Calendar** [1]) — wymyśliła grupa amerykańskich i zachodnioniemieckich informatyków. Pod przewodnictwem znanych grafików komputerowych, Horsta Helbiga z Instytutu Optoelektroniki w Wessling oraz Herberta W. Granke z Laboratorium Graficznego na University of Utah w Salt Lake City — na rok 1984 zaserwowali oni zestawione ze smakiem dwanaście zachwycających kolorystyką plansz.

Sama zasada przedstawienia wykazu dni na cały miesiąc ilustracjami nie jest niczym nowym. Ale zamiast zwykłych w takich razach ogólnych sformułowań, zachwalających mniej lub bardziej natarczywie określony wyrób lub usługi — specjalizujący się w literaturze naukowej „ten drugi Springer" uciekł się do oryginalnego chwytu bibliograficznego.

Otóż ów **Kalendarz Informatyczny** — wydany w języku angielskim i uzupełniony niemieckim skrótem — jest chyba pierwszym w świecie k a l e n d a r z e m, na łamach którego podane są źródłowe charakterystyki ponad 20 prac naukowych. Są to przede wszystkim oryginalne artykuły Forda i Fulkersona, opisujące ich słynny algorytm maksymalizacji przepływu w sieci oraz wiele nieoczekiwanych i wprost zaskakujących związków z innymi działami matematyki/informatyki.

Drugą grupę stanowią rekursyjne ciągi Thue'go, odkryte przezeń jeszcze na początku bieżącego wieku, a stanowiące przedmiot zainteresowania współczesnych matematyków działających w dziedzinie informatyki teoretycznej — np. Jacobsa, którego pracę na temat związków teorii ergodycznej z kombinatoryką wydawnictwo Springera opublikowało w ubiegłym roku.

Trzeci kompleks związany jest ze zbiorami juliańskimi, nazwanymi tak na cześć matematyka G. Julia (1893—1978), który zajmował się jeszcze w latach dwudziestych problemami tzw. atraktorów. Właśnie iteracje newtonowskie równań zmiennej zespolonej prowadzą do interesujących graficznie zbiorów juliańskich — w których płaszczyzna zamalowana jest trzema barwami w ten skomplikowany sposób, że żaden punkt nie ma dwubarwnego otoczenia....

Są jeszcze wykresy „poziomicowe" prostych zadawałoby się funkcji dwóch zmiennych, ale dających — przy pewnych wartościach parametrów — fenomenalne efekty wizualne (oglądane na kolorowym monitorze komputerowym). To wszystko zachwyca, zaciekawia, prowadzi do zadumy i chęci bliższego poznania szczegółów o systemie DIBIAS-II. System ten zawdzięcza swe istnienie konieczności graficznego opracowywania danych satelitarnych. Książka **Computergrafik-Computerkunst** wspomnianego Herberta W. Franke na ten temat i wynikających stąd związków ze sztuką komputerową ukaże się jeszcze w tym roku.

Aha, więc szydło reklamowe wyszło jednak z kalendarza! Choć właściwie jest to igiełka tylko skryta chytrze w stogu informacji. A ten „stóg" przypomina najlepsze tradycje popularyzatorskie **Kalejdoskopu Matematycznego** — właśnie łączącego przypisy źródłowe ze swobodną fabułą i operującego głównie obrazem, znanym zresztą szeroko z licznych przekładów z polskiego na różne języki obce. Aż żal, że nasi wydawcy nie potrafili w ten sposób wykorzystać dorobku popularyzatorskiego prof. Hugona Steinhausa. Trudno, samiśmy sobie winni...

Może jednak warto by się zastanowić nad polską wersją **Mathematics and Computing 1985?** Grafika mozaikowa — to przecież niekłopotliwy sposób mówienia o informatyce. Kiedy zaś poznamy smak nowego typu kalendarza, przyjdzie i czas na własne opracowania.

Dobry wydawca potrafi się inspirować nie byle czym, a na byle co nie może sobie pozwolić, bo dobra inspiracja rodzi szybko następne dobre inspiracje. A to jest jedna z ubocznych funkcji kalendarzy. Zwłaszcza informatycznych.

A.B.E.

[1]) Mathematics and Computing 1984. Kalendarz miesięczny pod redakcją Roberta Hollandera, Springer-Verlag, Nowy Jork 1983; 26 str. nlb; 13 plansz kolorowych; bibl. 24 poz.

KALENDARZ

Luty

14—17, Berlin Zachodni: VII kongres i wystawa sprzętu „Online 84" — organizator: Online GmbH

Marzec

11—17, Lipsk, ekspozycja sprzętu komputerowego, w ramach Wiosennych Targów Lipskich
12—15, Las Vegas (USA: XII konferencja „Interface '84" na temat transmisji i przetwarzania danych — organizator: The Interface Group, Inc.

19—21, San Diego (USA): międzynarodowa konferencja na temat przetwarzania mowy i sygnałów — organizator IEEE temat wydajności komputerowych systemów telekomunikacyjnych — organizator: IBM Zürich Research Laboratory
21—23, Zurych (Szwajcaria): międzynarodowe sympozjum na
26—28, Karlsruhe (RFN): konferencja na temat architektury i eksploatacji systemów komputerowych — organizator: Gesellschaft für Informatik

26—29, Orlando (USA): VII międzynarodowa konferencja na temat inżynierii oprogramowania — organizator: IEEE

Kwiecień

11—13, Paryż: STACS — konferencja na temat teoretycznych aspektów informatyki — organizator: AFCET oraz Gesellschaft für Informatik
17—19, Tuluza (Francja): VI międzynarodowe kolokwium na temat programowania — organizator: Universitè Paul Sabatier
23—25, Pitsburg (USA): konferencja na temat rozwoju oprogramowania użytkowego — organizator: ACM Sigsoft/Sigplan

Maj

14—17, Amsterdam (Holandia): międzynarodowa konferencja telekomunikacyjna IEEE — organizator: TACM

Czerwiec

4—6, Nicea (Francja): kolokwium na temat predyspozycji w programowaniu — organizator: AFCET
7—8, SophiaAntipolis (Francja): konferencja na temat jakościowej oceny predyspozycji w programowaniu — organizator: AFCET
17—21, Kopenhaga: XXVI międzynarodowa konferencja TIMS — organizator: The Institute of Management Sciences
26—29, Rzym: II światowa konferencja na temat międzynarodowego przepływu danych — organizator: Intergovernmental Bureau for Informatics

Cykl życia oprogramowania

Według normy IEEE Std. 729, cykl życia oprogramowania jest to okres od powzięcia decyzji o wytworzeniu produktu do wycofania produktu z użycia („the period of time that starts when a software product is conceved and ends when the product is no longer available for use"). Choć nie osiągnięto zgodności co do tego, na jakie fazy dzieli się cały cykl, wymienia się następujący, przykładowy podział:

— formułowanie wymagań
— projektowanie
— realizacja
— testowanie
— instalowanie i sprawdzanie
— eksploatacja i pielęgnowanie
— wycofanie produktu.

Faza formułowania wymagań (ang. requirements phase) jest to okres w cyklu życia oprogramowania, podczas którego definiuje się i dokumentuje wymagania dotyczące produktu programowego, np. związane z jego możliwościami funkcjonalnymi i wykonawczymi (ang. functinonal and performance capabilities). Istotną rolę odgrywają wymagania projektowe (ang. design requirements), które ograniczają lub wpływają na projekt systemu oprogramowania lub — jego składników. Wyróżnia się kilka rodzajów wymagań: funkcjonalne, wykonawcze, sprzężeniowe.

Wymagania funkcjonalne (ang. functional requirements) określają funkcje, które musi spełniać system lub jego składniki. Wymagania wykonawcze (ang. performance requirements) określają właściwości wykonawcze, które musi mieć system lub jego składniki (np. szybkość, dokładność, częstotliwość). Wymagania sprzężeniowe (ang. interface requirements) określają elementy sprzętu, oprogramowania lub bazy danych, z którymi musi być sprzężony system lub jego składniki, lub — narzucają ograniczenia na formaty, zależności czasowe lub inne właściwości związane z takim sprzężeniem.

Wymagania formułuje się w postaci tzw. specyfikacji. Ogólnie, specyfikacja jest to dokument opisujący — w pełny, precyzyjny i sprawdzalny sposób — wymagania, projekt, zachowanie lub inne charakterystyki systemu lub jego skałdników. Wyróżnia się specyfikacje wymagań projektowych, funkcjonalnych, wykonawczych i sprzężeniowych, a ich definicje są na ogół rozszerzeniem podanych wyżej definicji samych wymagań.

Faza projektowania (ang. design phase) jest to okres w cyklu życia oprogramowania, podczas którego tworzy się, dokumentuje i weryfikuje (pod względem spełnienia wymagań) projekty architektury, składników oprogramowania, sprzężeń i danych. Faza projektowania dzieli się na części, wśród których wyróżniono projektowanie wstępne i szczegółowe.

Projektowanie wstępne (ang. preliminary design) jest procesem polegającym na analizowaniu alternatywnych projektów i określaniu architektury oprogramowania. Na typowe projektowanie wstępne składa się definiowanie i tworzenie struktury składników i danych programu komputerowego, definiowanie sprzężeń i określanie wymagań czasowych oraz dotyczących wielkości systemu.

Projektowanie szczegółowe (ang. detailed design) jest procesem doskonalenia i rozszerzania projektu wstępnego w celu sformułowania bardziej szczegółowego opisu reguł przetwarzania, definicji i struktur danych, w takim stopniu, aby projekt był dostatecznie dojrzały do realizacji.

Ważnym wydarzeniem w fazie projektowania jest tzw. przegląd projektu (ang. design review), tj. formalne spotkanie, na którym użytkownikom, nabywcom lub innym zainteresowanym przedstawia się wstępny lub szczegółowy projekt systemu w celu zebrania uwag i zatwierdzenia.

Faza realizacji (ang. implementation phase) jest to okres, podczas którego tworzy się i uruchamia (ang. debug) produkt programowy na podstawie dokumentacji projektowej. Wymagania implementacyjne są to wymagania, które ograniczają lub wpływają na realizację projektu oprogramowania.

Faza testowania (ang. test phase) jest to okres, podczas którego ocenia się i integruje składniki produktu programowego oraz ocenia się sam produkt pod względem spełnienia wymagań. Niektóre pojęcia dotyczące testowania (a także pojęcie uruchamiania) omówiono bardziej szczegółowo w numerze 11/83.

Faza instalowania i sprawdzania (ang. installation and checkout phase) jest to okres, podczas którego integruje się produkt programowy z jego środowiskiem operacyjnym i testuje w tym środowisku w celu zapewnienia pożądanego działania.

Wszystkie wymienione fazy, niekiedy z pominięciem fazy instalowania i sprawdzania, nazywane są **cyklem rozwojowym oprogramowania.** Według określenia definicyjnego cykl rozwojowy oprogramowania (ang. software development cycle) jest to okres od powzięcia decyzji o opracowaniu produktu programowego do jego dostarczenia („the period of time that begins with the decision to develop a software product and ends when the product is delivered"). Dostawa (ang. delivery) jest to punkt cyklu rozwojowego oprogramowania, w którym produkt zostaje przekazany nabywcy w celu użytkowania. Ważną rolę w przekazaniu produktu odgrywają kryteria odbioru (ang. acceptance criteria), tzn. kryteria, którym musi odpowiadać produkt programowy, aby skutecznie przejść fazę testowania lub spełnić wymagania dostawy.

Istotą omówionego powyżej procesu rozwoju oprogramowania (ang. software development process) jest więc transformacja potrzeb użytkownika na wymagania programowe, na podstawie których tworzy się projekt i realizuje się go w postaci kodu programu. Kod ten testuje się, dokumentuje i zatwierdza do użytku.

W pełnym cyklu życia oprogramowania mieści się jeszcze faza eksploatacji i pielęgnowania (ang. operation and maintenance phase). Tym mianem określa się okres, podczas którego produkt programowy stosuje się w jego środowisku operacyjnym, nadzoruje pod względem wykonawczym (ang. satisfactory performance) i modyfikuje — zależnie od potrzeb — w celu poprawienia usterek lub dostosowania do zmienionych wymagań.

Do pełnego cyklu zalicza się czasami także fazę wycofania produktu (ang. retirement phase). Jest to okres, podczas którego kończy się pielęgnowanie produktu programowego.

JANUSZ ZALEWSKI

KSIĄŻKI NADESŁANE

WYDAWNICTWA NAUKOWO-TECHNICZNE

Kluźniak F., Szpakowicz S.: PROLOG. Seria BIO, nakład 4000 egz., 280 str., cena 160 zł. Warszawa. 1983.

Książka zawiera opis niekonwencjonalnego języka programowania PROLOG. Jego prezentacja rozpoczyna się od obszernego, intuicyjnego wprowadzenia w specyficzne dlań metody programowania. Dalej czytelnik znajdzie bardziej formalny i kompletny opis wersji PROLOGU, zrealizowanej na maszynie cyfrowej ODRA 1305. W następnych częściach książki autorzy zajmują się problematyką nieelementarnych struktur danych, przedstawiają szereg wskazówek metodycznych dotyczących zasad dobrego programowania w PROLOGU oraz prezentują praktyczne zastosowania tego języka.

Owczarczyk J.: Interakcyjny system przetwarzania obrazów

INFORMATYKA 1984, nr 2, s. 2

Ogólna charakterystyka struktury oraz zastosowań współczesnych interakcyjnych systemów rozpoznawania obrazów. Omówiono uwarunkowania efektywnej współpracy systemu z użytkownikiem oraz rozwiązania sprzętowe najważniejszych jego składników.

Овчарчык Й.: Интерактивная система обработки образов

INFORMATYKA 1984, № 2, стр. 2

Общая характеристика структур и применений современных интерактивных систем распознавания образов. Обсуждены обусловленности эффективного взаимодействия системы с пользователем и аппаратные решения ее основных частей.

Fuglewicz P.: MODULA-2 — język lat osiemdziesiątych (2). Opis składni

INFORMATYKA 1984, nr 2, s. 6

Druga część charakterystyki języka MODULA-2, zawierająca ogólny opis jego składni. Omówiono słownik języka, deklaracje, wyrażenia, instrukcje oraz jednostki kompilacji.

Фуглевич П.: MODULA-2 — язык восьмидесятых годов (2). Описание синтаксиса

INFORMATYKA 1984, № 2, стр. 6

Вторая часть характеристики языка MODULA-2 содержащая общее описание его синтаксиса. Обсуждаются словарь языка, объявления, выражения, операторы и сегменты компиляции.

Serafin E., Serafin W.: Programowy tester jednostek pamięci na dyskach elastycznych PLx45D.5

INFORMATYKA 1984, nr 2, s. 10

Charakterystyka programu zastępującego urządzenie testujące TSP-45A do sprawdzania poprawności działania i regulacji podstawowych zespołów i parametrów eksploatacyjnych jednostek pamięci na dyskach elastycznych.

Серафин Э., Серафин В.: Программный тестер памяти на гибких дисках PLx45D.5

INFORMATYKA 1984, № 2, стр. 10

Характеристика программы заменяющей испытывающее устройство TSP-45A для испытаний правильности действия и регулирования основных узлов и эксплуатационных параметров запоминающих устройств на гибких дисках.

Owczarczyk J.: Interactive image processing system

INFORMATYKA 1984, No. 2, p. 2

General characteristics of contemporary interactive image processnig systems structure and applications. Conditions for system's effective cooperation witn user, as well as hardware solutions of its main parts are discussed.

Owczarczyk J.: Ein interaktives Bildverarbeitungssystem

INFORMATYKA 1984, Nr. 2, S. 2

Eine allgemeine Charakteristik der Struktur und Anwendungen von modernen interaktiven Bildverarbeitungssystemen. Es wurden Vorbedingungen einer effektiven Zusammenarbeit des Systems mit Benutzer, sowie Hardwarelösungen seiner wichtigsten Bestandsteile besprochen.

Fuglewicz P.: MODULA-2 — a language for 80's (2). Syntax description

INFORMATYKA 1984, No. 2, p. 6

Second part of MODULA-2 characteristics, which includes general description of its syntax. Language's vocabulary, declarations, expressions, statements and compiling units are discussed.

Fuglewicz P.: MODULA-2 — eine Sprache der achtzigen Jahre (2). Syntaxbeschreibung

INFORMATYKA 1984, Nr. 2, S. 6

Zweiter Teil einer Charakteristik von MODULA-2 — Programmiersprache, der eine allgemeine Beschreibung ihrer Syntax umfasst. Es wurdn Sprachewörterbuch, Deklarationen, Ausdrücke, Anweisungen und Kompilationseinheiten besprochen.

Serafin E., Serafin W.: Program tester for PLx45D.5 diskette storage units

INFORMATYKA 1984, No. 2, p. 10

Characteristics of the program, which replaces the TSP-45A testing device for diskette storage units checking and regulation.

Serafin E., Serafin W.: Programtester für PLx45D.5 Disketten-Plattenspeicher

INFORMATYKA 1984, Nr. 2, S. 10

Eine Charakteristik des Programmes, das die TSP-45A Testeinrichtung zur Prüfung der Funktionsordnungsmässigkeit und Regelung von Disketten-Plattenspeicher ersetzt.

robotron

Podczas realizacji złożonych, wieloprzekrojowych zadań przetwarzania danych, jakie występują w kombinatach i zakładach przemysłowych, w administracji publicznej, w transporcie i komunikacji, czy w instytutach naukowych i biurach projektowych, ujawniają się takie zalety tego komputera, jak:

- małe zapotrzebowanie miejsca oraz energii w wyniku zastosowania układów o dużym stopniu scalenia i nowoczesnych rozwiązań zasilania
- zastosowanie różnych metod zdalnego przetwarzania danych oraz możliwość użycia w sieci komputerowej jako systemu sterującego
- znaczna pojemność pamięci operacyjnej (do 4096 K bajtów)
- możliwość efektywnego wykorzystywania obszaru pamięci operacyjnej do pojemności 16 M bajtów w wyniku zrealizowania koncepcji pamięci wirtualnej

EC 1055 M

nasz nowy komputer do typowych zadań przetwarzania danych

- możliwość eksploatacji zarówno w rozszerzonym trybie sterowania zgodnie z koncepcją komputerów serii RIAD 2, jak i w podstawowym trybie sterowania,

EC 1055 M

zgodnie z koncepcją komputerów serii RIAD 1.

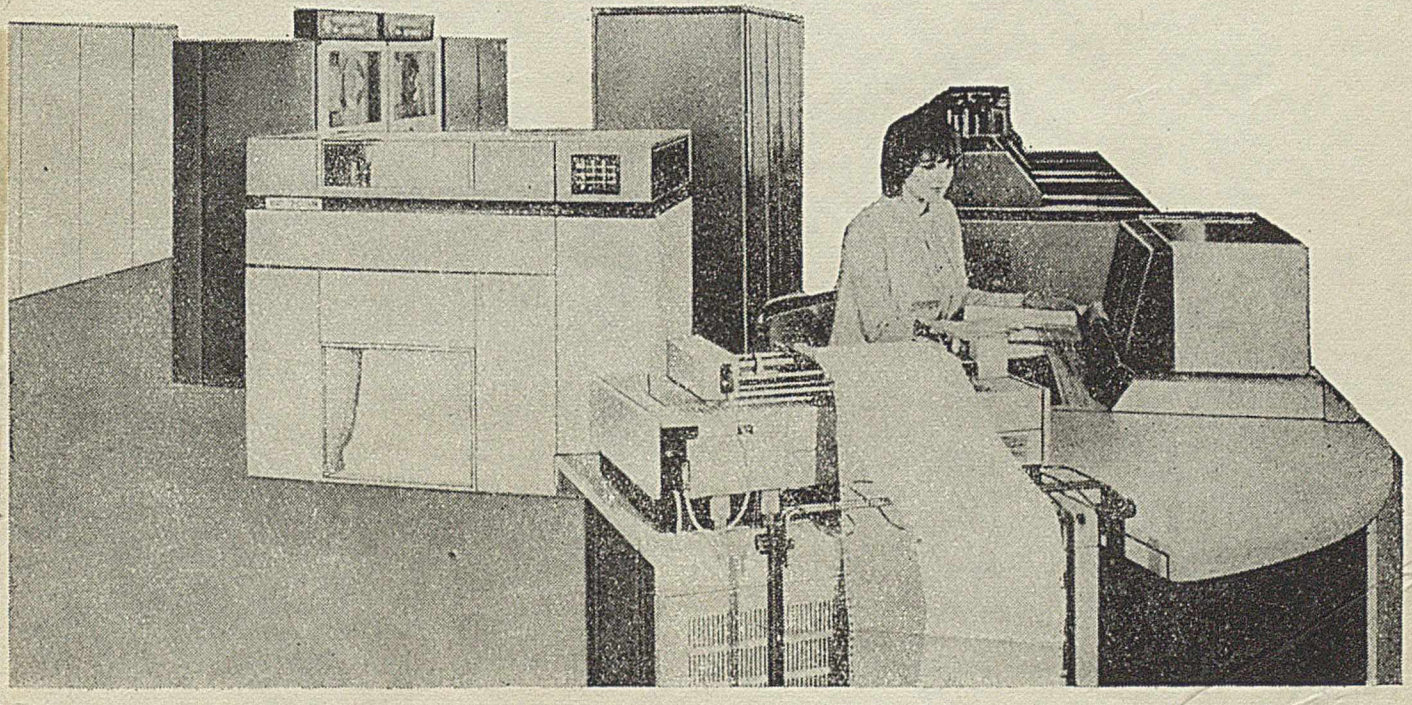
Najbardziej interesującą i efektywną innowacją jest system operacyjny SVM/ES oparty na koncepcji maszyn wirtualnych. Każda z takich maszyn stanowi dla użytkownika niezależny komputer z własnymi zasobami, działający pod nadzorem własnego, dobranego zgodnie z potrzebami, systemu operacyjnego.

Dzięki tej bardzo efektywnej koncepcji pojawia się wiele nowych możliwości zastosowań robotrona EC 1055 M, warto więc zgłosić się do nas o szczegółową dokumentację tego komputera.

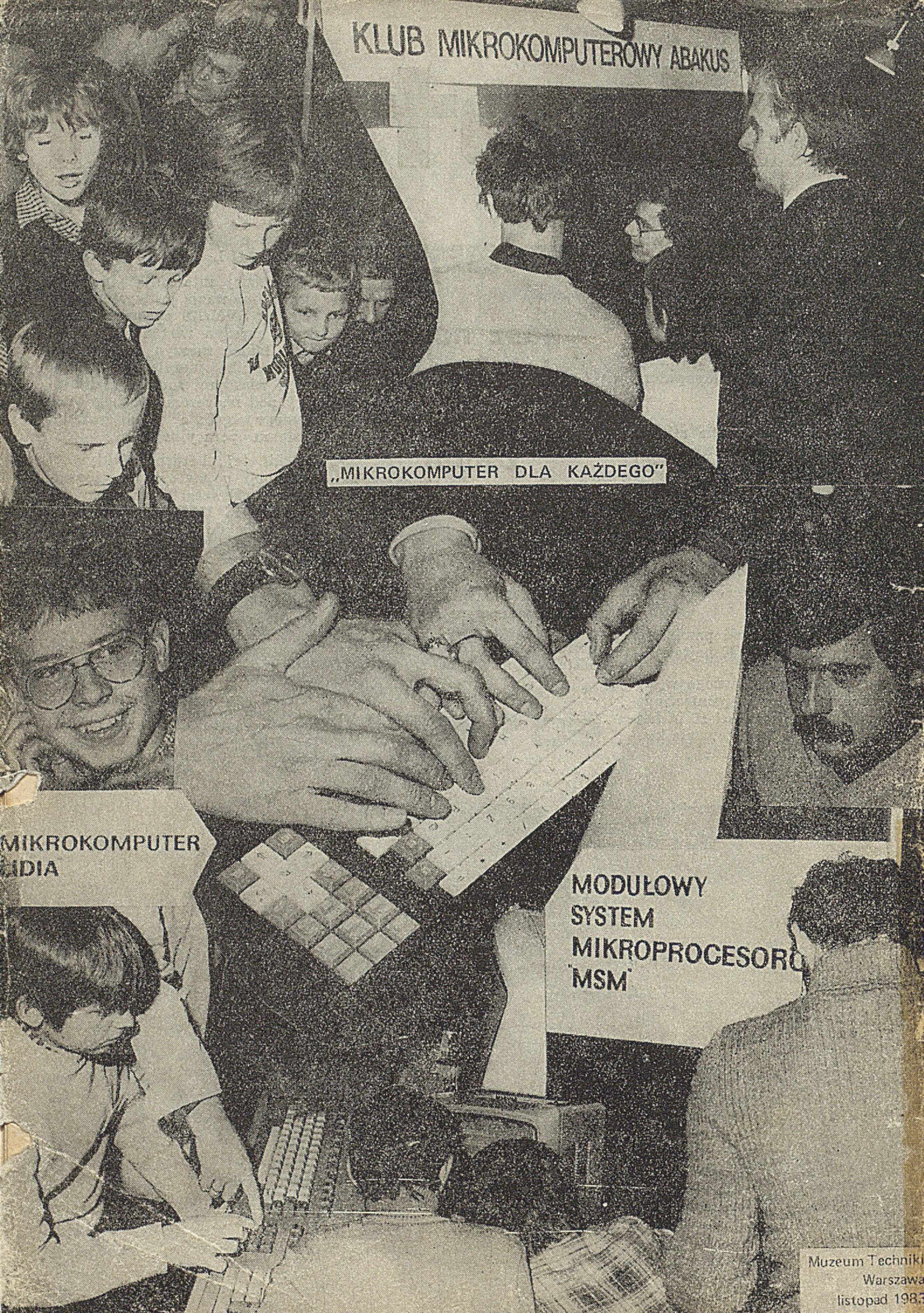
robotron

Przedsiębiorstwo Handlu
Zagranicznego NRD
Robotron Export-Import
DDR — 1080 Berlin
Friedrichstrasse 61

Prosimy odwiedzić nas w hali nr 15 podczas Wiosennych Targów Lipskich w czasie od 11 do 17 marca 1984 oraz w biurze eksportowo-importowym w hali nr 12.12



EO/1211/K/83-A



Muzeum Techniki
Warszawa
listopad 198